Recent Thermal Management Techniques for Microprocessors

JOONHO KONG, SUNG WOO CHUNG
Korea University, Seoul, Korea
AND
KEVIN SKADRON
University of Virginia, Charlottesville, VA

Microprocessor design has recently encountered many constraints such as power, energy, reliability and temperature. Among these challenging issues, temperature-related issues have become especially important within the past several years. We summarize recent thermal management techniques for microprocessors, focusing on those that affect or rely on the microarchitecture. We categorize thermal management techniques into six main categories: temperature monitoring, microarchitectural techniques, floorplanning, OS/compiler techniques, liquid cooling techniques, and thermal reliability/security. Temperature monitoring – a requirement for dynamic thermal management (DTM) – includes temperature estimation and sensor placement techniques for accurate temperature measurement or estimation. Microarchitectural techniques include both static and dynamic thermal management techniques that control hardware structures. Floorplanning covers a range of thermal-aware floorplanning techniques for 2D and 3D microprocessors. OS/compiler techniques include thermal-aware task scheduling and instruction scheduling techniques. Liquid cooling techniques are higher-capacity alternatives to conventional air cooling techniques. Thermal reliability/security issues cover temperature-dependent reliability modeling, dynamic reliability management (DRM), and malicious codes that specifically cause overheating. Temperature-related issues will only become more challenging as process technology continues to evolve and transistor densities scales up faster than power per transistor scales down. The overall objective of this survey is to give microprocessor designers a broad perspective on various aspects of designing thermal-aware microprocessors and to guide future thermal management studies.

Categories and Subject Descriptors: C.5.3 [**Computer System Implementation**]: Microcomputers—*Microprocessors*; C.5.4 [**Computer System Implementation**]: VLSI Systems; D.4.1 [**Operating Systems**]: Process Management—*Scheduling*;

General Terms: Design, Management

Additional Key Words and Phrases: Thermal management, microprocessor, performance and reliability

ACM File Format:
KONG, J., CHUNG, S. W., AND SKADRON, K., 2010. Recent Thermal Management Techniques for Microprocessors. ACM Comput. Surv.,

Authors' addresses: J. Kong and S. W. Chung, Department of Computer Science and Engineering, Korea University, Seoul, Korea. E-mail: {luisfigo77, swchung}@korea.ac.kr; K. Skadron, Department of Computer Science, University of Virginia, Charlottesville, VA. E-mail: skadron@cs.virginia.edu

1. INTRODUCTION

Using performance as the primary objective in microprocessor design has led to increasingly sophisticated processor organizations such as superscalar, out-of-order issue, and Very Long Instruction Word (VLIW). In addition to such microarchitectural techniques, process technology continues to double the number of transistors per unit area every 1.5~2 years by reducing feature sizes and shrinking the distance between devices, allowing continued increases in operating frequency as well as supporting ever more sophisticated processor organizations within a given area or cost budget. Unfortunately, these performance improvements have come at increasingly high costs in power and cooling requirements. As processor structures become more sophisticated and operate at higher frequency, they dissipate more power per unit area and the processors overall dissipate more total power. Both power density and total power have increased steadily over the last 25 years. Even if microarchitectural complexity stops increasing, power densities will still rise, because supply voltage is no longer scaling down as fast as feature size [SIA 2009]. At the same time, we are approaching the limits of air cooling (around 150~200W) [SIA 2009], and no mass-market alternatives have become apparent. Clearly, this growth in power dissipation cannot be sustained. Of course, the shift to multi-core architectures has temporarily alleviated the problem, as performance can now be achieved through parallelism and microarchitectural complexity has indeed slowed or even reversed. However, as more cores are integrated and supply voltage scaling slows even more, power densities will again become a severe challenge.

High temperatures pose a reliability challenge, since many aging mechanisms, such as electro-migration and dielectric breakdown, are exponentially dependent on temperature. Inadequate thermal control can lead to complete failure, as several recent products have shown (e.g., [ARS Technica 2008; EE Times 2008]). Moreover, temperature must be addressed from the *earliest* design stages, because early design choices, such as the number and complexity of cores, dictate the basic activity patterns that a processor will exhibit [Li et al. 2006]. As a result, microprocessor architects have begun to study thermal management more carefully in the early-stage of design-time. In order to design efficient thermal management techniques, accurate thermal analysis at the design-time is essential. However, thermal analysis in the design stage is not an easy process. The reasons are as follows: 1) Temperature is typically represented by complex non-linear equations (e.g., Newton's law of cooling or Fourier's law), making thermal analysis for microprocessors in the design-time difficult without complex as well as accurate simulations. Furthermore, due to its complexity, temperature simulations typically need quite longer time than other metrics such as performance and power. 2) The RC thermal time constant is heavily dependent on environmental parameters such as material and packaging [Mesa-Martinez et al. 2010]. Thus, the designers should be careful when analyzing a thermal behavior since temperature response can be different according to the various environmental parameters. It calls for simple and accurate simulation tools for microprocessor designers.

To support early-stage thermal-aware microprocessor design, several temperature simulation tools have been developed. These include HotSpot [Skadron et al. 2003; Huang et al. 2008] and ATMI [Michaud and Sazeides 2007]. These rely on compact models for the heat transport, and generally represent each microarchitectural unit with a uniform power density. They also rely on some external source for the power dissipation

in each unit. Analytical models or empirical values extracted and scaled from prior hardware (coupled with cycle-accurate microarchitectural simulations to obtain activity factors) are the most common sources. Using such early-stage-design simulation tools, hardware support for thermal management can be explored without the need for traditional but computationally expensive thermal modeling tools based on finite-difference or finite-element methods, or empirical measurements using hardware. However, note that thermal modeling is beyond the scope of this paper. We restrict the main topic of our survey only to thermal management techniques, excluding thermal modeling techniques.

Power management can help control temperatures, because reducing power dissipation may also reduce power density. However, reducing power alone is not always effective and may in fact conflict with thermal management: if power reductions are achieved by turning off under-utilized structures and concentrating activity in a smaller area, power density actually increases! Conventional power-saving techniques also tend to exploit slack when the processor is under-utilized and power-saving techniques will typically have less impact on performance [Venkatachalam and Franz 2005], while thermal management is chiefly a concern when the processor is heavily utilized and power-saving techniques may severely affect performance. Power-saving techniques used for managing current delivery or energy efficiency may also target structures that are not hot, and hence have limited impact on temperature. Furthermore, temperature changes slowly compared to the potential rate of change of activity within processor structures—over microseconds or longer—because even a small silicon die contains significant material mass. This means that power management will affect temperature only if the power reduction is maintained for a sufficiently longer time. For these reasons, even though power management for power delivery or energy efficiency concerns may use the same techniques as for thermal management, the *policies* may be different and even potentially in conflict.

In this paper, we introduce recent thermal-aware microarchitecture techniques. We restrict our survey to the temperature-related studies, excluding studies which only consider power (or energy). Note we do not provide all of the evaluation results under the identical evaluation framework since the evaluation environments (such as packaging, cooling, and processor parameters) of the studies vary. The rest of this paper is organized as follows. Section 2 provides a brief organization of our survey. Section 3 introduces recent representative temperature management techniques. In the last section, we conclude our paper.

2. A HIERARCHICAL ORGANIZATION OF OUR SURVEY

Prior to conducting a detailed survey, we categorize our survey hierarchically into six main parts; temperature monitoring, microarchitectural techniques, floorplanning, OS/compiler techniques, liquid cooling techniques, and thermal reliability/security. We summarize the organization of our survey as depicted in Fig. 1. The *Temperature monitoring* section discusses how to monitor the temperature of microprocessors effectively and accurately, and breaks down into two sub-sections. In the *Temperature estimation* sub-section, we introduce many cost-effective and accurate temperature estimation techniques, and in the *On-chip sensor placement* sub-section, we introduce

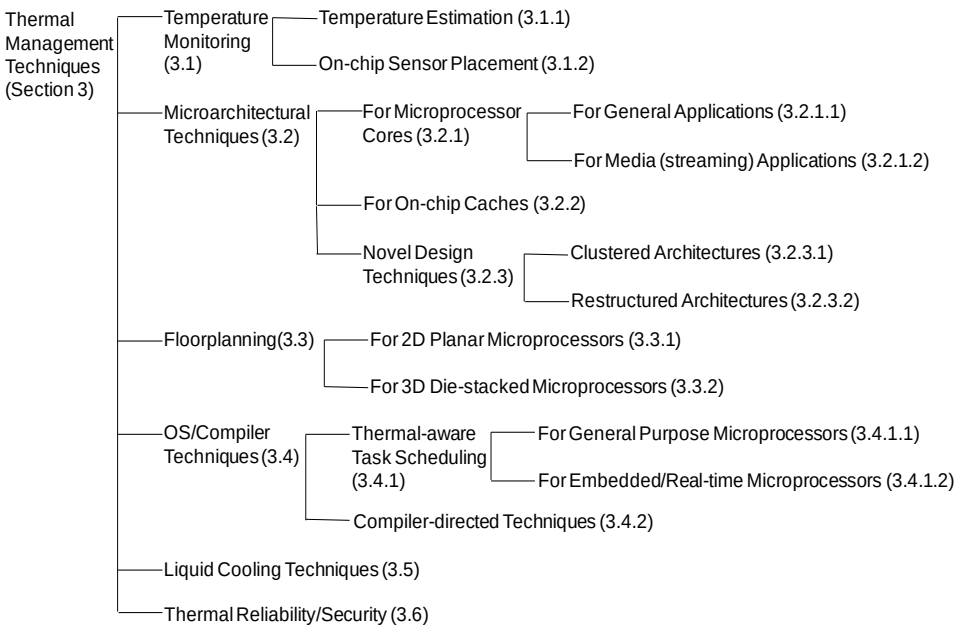


Fig. 1 A hierarchical organization of our survey (the number in the parentheses indicates the section number)

several techniques which take into account the appropriate allocation of thermal sensors. The *Microarchitectural techniques* section is composed of three sub-sections. In the *For microprocessor cores* sub-section, we introduce recent outstanding microarchitectural thermal management techniques for microprocessor cores. The next sub-section, *For On-chip caches*, introduces several techniques to reduce power density of on-chip caches by spreading heat while maintaining locality. In the *Novel design techniques* sub-section, we introduce microprocessor structures to reduce the power density of the microprocessor by dividing it into several clusters or restructuring the typical microprocessor architectures. The *Floorplanning* section summarizes recent floorplanning techniques from both CAD and microarchitectural perspectives. The *For 2D planar microprocessors* and *For 3D die-stacked microprocessors* sub-sections introduce thermal-aware floorplanning techniques for 2D and 3D microprocessors, respectively. In the *OS/compiler techniques* section, we explore many temperature management techniques orchestrated by OS or compilers. The *Thermal-aware task scheduling* sub-section discusses how to schedule and assign the tasks (processes or threads) to cores for managing the temperature of microprocessors. The *Compiler-directed techniques* sub-section introduces thermal management techniques based on compiler optimizations. Several thermal-aware code optimization techniques are introduced here. In the *Liquid cooling techniques* section, we investigate liquid cooling techniques for microprocessors that are outstanding alternatives to conventional air cooling, and evaluate the effectiveness of liquid cooling in the computer architectural-level. The last section, *Thermal reliability/security*, discusses thermal-aware reliability modeling techniques, reliability improvement techniques, and several techniques for thermal security.

3. THERMAL MANAGEMENT TECHNIQUES

3.1. Temperature Monitoring

Prior to introducing temperature management techniques, we first introduce temperature monitoring techniques, which are crucial for efficient dynamic thermal management. For example, the IBM Power7 microprocessor [Ware et al. 2010] employs 44 digital thermal sensors on a chip. Though such a large number of thermal sensors might be considered excessive, Power7 aims to capture almost all the corner cases that could not detected by a smaller number of thermal sensors. This implies that the large number of thermal sensors is worthwhile for accurate temperature monitoring.

3.1.1. Temperature Estimation

Temperature sensing or estimation is important, since DTM (Dynamic Thermal Management) uses the current temperature as feedback to adjust its control and force further reductions in temperature or permit higher performance, as appropriate. Thus, the thermal sensor or temperature estimation should be cost-efficient but measure on-chip temperature accurately. Note that a thermal sensor has its own power and area cost and temperature estimation also has run-time performance overheads as well as power overheads.

Typically, there are two types of thermal sensors used for thermal sensing of microprocessors: digital and analog thermal sensors. An analog thermal sensor is mainly composed of ring oscillators. It utilizes the fact that CMOS inverter delay depends on the temperature. On the other hand, a digital thermal sensor utilizes a band-gap voltage reference circuit whose output voltage is dependent on the temperature. In recent microprocessor design, due to higher accuracy and smaller area overhead, digital thermal sensors are more preferred than analog thermal sensors. In practice, digital thermal sensors are mainly used to detect *localized hotspots*, while analog thermal sensors are used to read *on-die temperatures* [Naveh et al. 2006]. Many robust and accurate thermal sensor design techniques have been proposed so far [Chen et al. 2005; 2006; Remarsu and Kundu 2009; Zhang and Srivastava 2009]. However, they are out of scope of this paper, since thermal sensor design techniques themselves are not microarchitectural techniques. Thus, we limit the scope of this sub-section only to the exploration of temperature estimation techniques, not including thermal sensor design.

For more accurate thermal measurements under process variation, there have been many studies using *model-based* temperature estimation. The main advantage of model-based temperature estimation is that it is more robust to severe noise or process variation, in contrast to relying on temperature sensors that are also vulnerable to noise or process variation. The model-based approach is shown to successfully compensate for inaccurate thermal sensing. Jung and Pedram [2008] proposed a Kalman-filter-based temperature and power estimation technique, based on junction temperature and the power state. The junction temperature estimation uses a Kalman filter [Kalman 1960], while the power estimation is based on POMDP[1] [Puterman 1994]. Another technique using the Kalman filter was proposed by Sharifi et al. [2008]. Their first step is an off-line step which

[1] POMDP stands for *Partially Observable Markov Decision Process*, which is one of generalizations of Markov Decision Process.

utilizes a thermal model to generate a steady-state Kalman filter by calibrating the model with previously-gathered, imprecise temperature sensing and power estimation results. Then, by using the steady-state Kalman Filter in the on-line step, temperature measurement is accurate despite inaccurate temperature sensing from actual sensors. This technique reduces sensor reading error by 3.03ºC, on average, with little run-time performance overhead.

Kursun and Cher [2008] proposed a performance-counter-based temperature estimation technique for multi-core systems that also reflects process variation. In an off-line phase, a variation map is generated by analyzing chip information such as the results of a thermal stressmark or sensed temperature. Since process variation (including within die and die-to-die variation) may vary the process parameters in each core and processor, utilizing variation maps leads to more accurate temperature estimation. By applying this temperature estimation technique to a thermal-aware task-scheduling technique, tasks are assigned to cores in a temperature-aware fashion (e.g., hot tasks are assigned to cool cores and cool tasks are assigned to hot cores). As a result, the temperature with the variation-aware technique is reduced by 4.5ºC compared to the variation-unaware technique.

In another temperature estimation technique considering process variation, Jaffari and Anis [2008] proposed a statistical method. Process variation induces severe leakage variation. With the conventional deterministic temperature/power estimation, process variation may lead to inaccurate temperature estimation because this leakage variation may make power and hence temperature estimation results inaccurate. For example, actual power consumption of equivalent Functional Units (FU) may be different across chips even with the same access rate to the FUs. Moreover, this inaccuracy may be more serious due to temperature-leakage dependence. In the proposed technique, the chip area is divided into a grid. Leakage power variability is modeled as a statistical function. The expected value of the temperature is found using this probability density function, whose inputs are dynamic power consumption and the package model. As a result, the proposed probabilistic method presents less than 0.3% error in temperature estimation compared to Monte Carlo simulation results.

In addition to CMOS thermal sensors, there have been studies on thermal sensing using software techniques. Localized temperature measurement is difficult because thermal sensors cannot detect localized hotspots that are far away from the limited number of sensor locations. To cover the entire chip area, the chip designer may deploy many thermal sensors. However, deploying many CMOS thermal sensors incurs area and power overhead. To detect unmonitored localized hotspots with minimal hardware thermal sensor overhead, a novel software thermal sensor was proposed by Chung and Skadron [2006a]. Their proposed technique is based on the performance-counter-based temperature estimation proposed by Lee and Skadron [2005]. Their run-time temperature estimation model uses the HotSpot tool and the run-time power model from Isci and Martonosi [2003]. However, in order to calculate the localized temperatures using Lee and Skadron's model, microprocessors should solve 4th order differential equations (the Runge-Kutta method in the HotSpot thermal model), and this is too complex to be executed at run-time. In order to avoid heavy computations, Chung and Skadron's technique uses a simple regression method for sensing localized temperatures. A simple linear equation, $Y=aX+b$ is used, where X is the performance counter value (the access rate of the functional unit) and Y is the temperature of the functional unit. The constants a

and *b* are determined through a simple regression analysis at design-time. Chung and Skadron's technique shows reasonable accuracy (at most 2.4ºC of the temperature difference with the peak temperature in the integer register file) with significantly low overhead. This work was evaluated using Dynamic Voltage and Frequency Scaling (DVFS) with off-line traces [Chung and Skadron 2006b] and it was actually adopted for the DVFS technique in the Intel Core2 Duo microprocessor by Lee et al. [2010]. The proposed technique detects and avoids most thermal emergencies by predicting localized hotspots with negligible performance overhead (mostly below 1%).

Khan and Kundu [2008] proposed a general software framework for predictive temperature management. While reactive thermal management techniques are sometimes inefficient due to the late response, their predictive technique eliminates this inefficiency. The proposed Virtual Thermal Manager (VTM) manages temperature by collaboration with hardware and software. The VTM maintains a Temperature History Table (THT) to predict near future temperatures. Each history entry corresponds to live threads in the system. Based on the history of latest temperatures and information of sensor readings, the predicted temperature is generated with using linear approximation. The hardware mechanism supports DTM by throttling Instruction-Level Parallelism (ILP) – for example, by adjusting issue width, retire width or speculation control. There are nine hierarchical severity levels which have different instruction bandwidths by throttling ILP. Considering the predicted temperature from the VTM, the last two temperature readings, the average temperature, and the last DTM action, the current DTM action (the severity level) is determined. The proposed technique improves the performance by 45% compared to DVFS and has almost identical thermal and performance efficiency to the control-theoretic technique [Skadron et al. 2002] (this technique will be discussed in Section 3.2.1.1), with a smaller hardware overhead and DTM response time.

3.1.2. On-chip Sensor Placement

We have discussed *how* to accurately sense (estimate) the temperature of microprocessors. However, sensor location is just as important as sensing (estimation) method in monitoring localized temperatures, since faraway thermal sensors are not sufficiently accurate. To measure localized temperatures accurately, we have to deploy many thermal sensors, which may be too costly. To resolve this problem, many researchers have studied efficient location of the limited number of the thermal sensors.

Gunther et al. [2001] introduced a thermal sensor placement on the Intel Pentium 4 microprocessor. By analyzing extracted thermal maps, they found optimized locations of the thermal sensors by identifying common hotspots. A thermal map is extracted by observing thermal behaviors of popular applications. To cover the entire chip area with the limited number of sensors, they set a thermal guard-band. This thermal guard-band ensures that no part of the microprocessor is in thermal emergency as long as the sensed temperature does not go over the threshold. Lee et al. [2005] proposed an analytical model that defines the magnitude of required guard-band. According to their model, temperature sensing gets less accurate as the distance between the heat source and the thermal sensor increases. Their analytical model fits well on the Intel Pentium 4, and they showed that the sensor placement and thermal guard-band setting of the Pentium 4 are appropriate.

Memik et al. [2008; Murkherjee and Memik 2006a] proposed a thermal sensor placement technique which considers thermal hotspots when executing general benchmark applications (SPEC2000). Utilizing the k-means clustering algorithm, with *k* sensors (clusters) and *n* hotspots (data points), the proposed technique finds the optimal point of each *k* (thermal sensors). Both global and local placements are considered for optimizing the sensor locations. The global placement is a chip-wide placement which considers well-known hotspots in the microprocessor. The local placement is at the component level, which is required for more fine-grained thermal optimizations (e.g., thermal-aware register banking). While their technique originally deploys a thermal sensor to each functional unit, it also provides a trade-off between the number of thermal sensors and temperature sensing accuracy through a hybrid technique which can deploy multiple thermal sensors in one functional unit. The proposed technique reduces the thermal sensing error down to 1.63ºC (3.18ºC at the worst case).

To sense the localized temperature with the limited number of thermal sensors, an interpolation scheme and interpolation-based dynamic selection method were proposed [Long et al. 2008; Memik et al. 2008]. In grid-based uniform sensor placement techniques, inaccurate temperature measurement may be carried out if the thermal sensor is far away from the hotspot. With collaborations among thermal sensors around each hotspot, the proposed interpolation scheme enables accurate temperature estimation of hotspots where thermal sensors are not deployed. An interpolation-based dynamic selection method is used to find localized hotspots with minimal sensor activation. With a large scale of processing data from many grid thermal sensors, there are serious interconnect power overheads due to heavy communication overheads. To reduce the large scale of data from many grid thermal sensors, the number of enabled thermal sensors should be minimized. To optimize the number of enabled thermal sensors, the interpolation-based dynamic selection method forms a hierarchy of the thermal sensors, as depicted in Fig. 2. First, coarse-grained thermal sensors (s0~s15 in Fig. 2) measure the temperature of each sensor location. Four adjacent thermal sensors deployed near the hottest spot (s5, s6, s9, and s10 in Fig. 2) are selected and nine grid thermal sensors within the selected four thermal sensors are activated. Among the estimated temperatures with nine grid sensors, the sensor location which shows the highest temperature is a hotspot. This technique enables a reduced number of the activated thermal sensors by a hierarchical usage while maintaining reasonable accuracy.

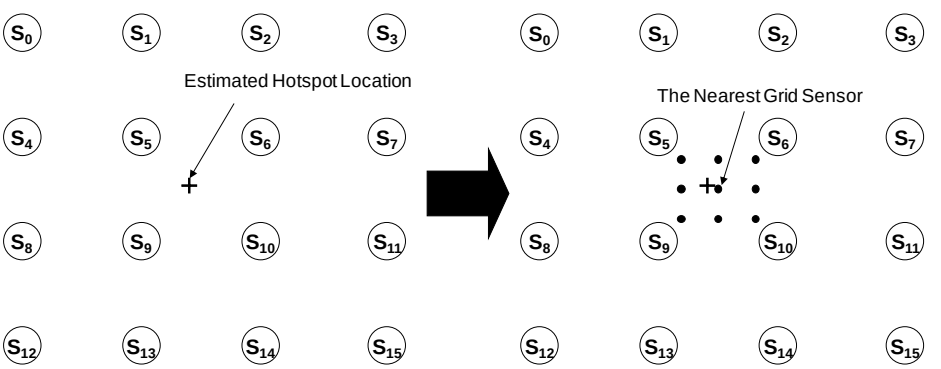


Fig. 2 An interpolation-based dynamic selection technique [Long et al. 2008; Memik et al. 2008]

3.2. Microarchitectural Techniques

3.2.1. For Microprocessor Cores

3.2.1.1. For General Applications

As predicted by Borkar [1999], temperature problems in microprocessors have become severe as process technology scales down feature sizes. Even though technology scaling allows higher clock frequencies and more sophisticated microarchitectures that make microprocessors faster and more power-efficient, they also increase power density. Even if clock frequencies and microarchitectural complexity abate, a growing problem is that supply voltage for full performance is no longer scaling down as fast as device feature size is being reduced [SIA 2009], and this leads to growing power densities at full performance. Although the operating supply voltage can be decreased by low power techniques such as DVFS, such techniques reduce performance and do not alleviate the growing severity of thermal stress at the rated performance.

Thermal problems have been addressed by many researchers since the early 2000s and thermal management is still an active research area. One representative thermal management research area is Dynamic Thermal Management (DTM). Early studies in DTM focused only on temperature management, ignoring performance optimization. The basic insight was to design the cooling solution for the worst *expected* power dissipation rather than a theoretical worst case with the maximum possible power dissipation. To protect against unexpected or malicious behaviors that exceed the capacity of such a cooling solution, temperature should be monitored and thermal excursions must throttle down the processor's activity and hence power dissipation. Initially, simplistic and costly throttling was used; for example, it is known that the Intel Pentium 4 simply reduces the duty cycle on the clock, and hence the overall activity of the processor, by 50% [Intel 2002].

If the *thermal design power* or TDP — the power dissipation that the thermal solution is designed to accommodate — is high enough, thermal excursions are unlikely and performance with throttling engaged is unimportant. However, as power dissipation and cooling costs continue to grow, more aggressive reductions in cooling costs become appealing. This increases the likelihood that DTM may engage during a legitimate application, and performance optimization becomes important to avoid the inevitable performance loss caused by DTM. Brooks and Martonosi [2001] evaluated the

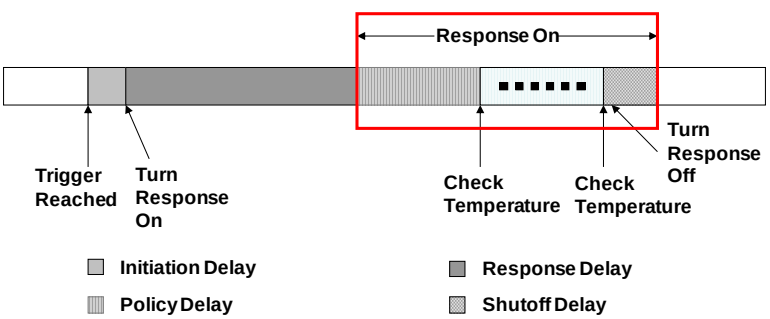
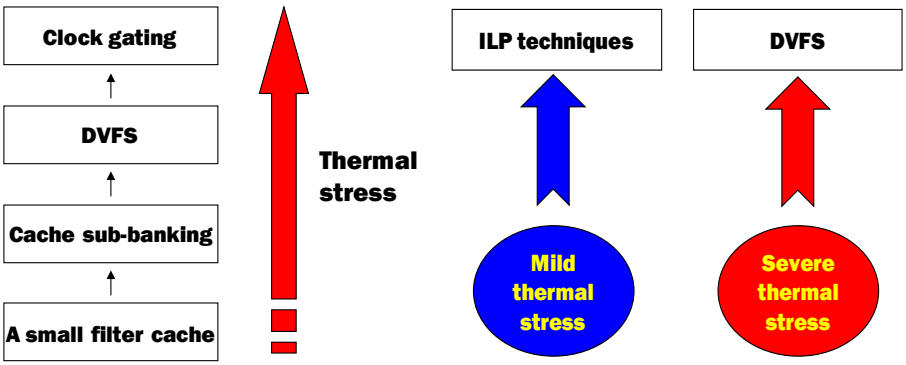


Fig. 3 DTM mechanisms [Brooks and Martonosi 2001]

performance impact of many DTM techniques for high performance microprocessors. They proposed DTM triggering, response, and initiation mechanisms focusing on reducing performance loss. Fig. 3 illustrates DTM mechanisms for microprocessors. When the temperature of the microprocessor reaches the pre-defined trigger temperature, there is an initiation delay before triggering DTM. After the DTM response is engaged, the microprocessor checks the temperature at each time interval. When the sensed temperature drops below the DTM trigger temperature, the DTM is disengaged and the microprocessor runs normally again. Disengagement may also incur some delay. Their proposed DTM response mechanisms can be categorized into voltage/frequency scaling and processor rate throttling. Clock frequency scaling and voltage/frequency scaling techniques adjust the clock frequency and/or voltage of the microprocessor *dynamically*. The relationship among dynamic power consumption, clock frequency, and supply voltage is given by $P \propto V_{dd}^2 f$, and we can approximate performance loss as being proportional to clock frequency. This means that frequency scaling alone, as well as voltage and frequency scaling together (DVFS, or dynamic voltage and frequency scaling), are both able to reduce temperature. Another approach proposed by Brooks and Martonosi is the use of microarchitectural techniques to throttle the instruction bandwidth of the microprocessor (thus throttling ILP and these techniques can also be referred to as *ILP techniques*). Note that DVFS is able to achieve a cubic reduction in power relative to the performance loss, while ILP throttling techniques generally only achieve linear reductions in power. Decode throttling and speculation control restrict the decode width and the number of unresolved branches, respectively. I-cache toggling turns off the front-end of the microprocessor, so that in-flight instructions continue but no new instructions are fetched. These all have the effect of reducing switching power, but like frequency scaling, achieve only linear reductions in power relative to the performance loss. An important distinction, however, is that ILP throttling techniques can be engaged with much lower latency than a change in the voltage and clock frequency. Actually, in their study, DVFS invocation overhead is assumed to be 10~50us which are much larger than the invocation overhead of ILP throttling techniques. In case of commercial processors such as Pentium M, DVFS invocation overhead is also 10~20us [Intel 2003]. Recently, however, DVFS invocation overhead (approximately ~5us) became much smaller [Intel 2010] due to the advance of control circuitry, which is still larger than ILP throttling overhead though.

To efficiently alleviate performance losses due to thermal management, hybrid and hierarchical DTM techniques were proposed. Fig. 4 is a conceptual structure of the hierarchical and hybrid DTM. The hierarchical DTM uses a gradual response mechanism to minimize the performance loss caused by the fixed DTM response [Huang et al. 2000]. The fixed DTM response applies the same DTM technique to different thermal stresses. Hence, it may over-react, and consequently incur excessive performance losses. On the other hand, the hierarchical DTM technique selects one out of four thermal management techniques (a small filter cache, data cache sub-banking, DVFS, and clock gating). The hierarchical DTM selects a more aggressive thermal management technique in case of thermal emergency, resulting in lower performance loss compared to using only clock gating or four techniques at once (*combined DTM*). A challenge with the hierarchical technique, however, is to select the correct mix of techniques. The hierarchical DTM technique shows much better performance than the combined DTM under various



(a) Hierarchical DTM [Huang et al. 2000] (b) Hybrid DTM [Skadron 2004]

Fig. 4 Hierarchical DTM and hybrid DTM

thermal constraints.

The hybrid DTM technique uses several response mechanisms to adaptively respond to thermal stresses for better performance [Skadron 2004]. This differs from the hierarchical DTM technique, because the response mechanism can be changed to achieve optimal performance, notably when thermal stress is not severe. For instance, in case of mild thermal stress, a DTM controller selects microarchitectural throttling techniques with low initiation delay, such as decode throttling or I-cache toggling, and further uses a feedback controller to select the duty cycle on this toggling mechanism. Conversely, in case of severe thermal stress (though not yet a thermal emergency), the controller engages DVFS, which has higher initiation delay but once engaged, achieves much better reductions in power for a given performance penalty. The key challenge in designing the hybrid DTM technique is to determine the proper balance between ILP techniques and DVFS. Compared to the conventional DVFS, the hybrid DTM reduces performance overheads (incurred by DTM) by 25%, on average.

Feedback control can be applied to most DTM techniques to minimize performance loss by adapting the specific setting (voltage/frequency, throttling duty cycle, etc.) to the minimal level required to maintain a safe temperature, and by adapting the setting as application behavior changes [Skadron et al. 2002; 2003]. PI control appears to suffice [Skadron et al. 2003; 2004]. Open-loop response, on the other hand, should use an aggressive response to ensure that temperature is controlled no matter how severe the stress, and as a result, the system is likely to over-react and performance then suffers unnecessarily. In one study by Skadron et al. [2002], closed-loop control reduces the performance loss by 65% on average compared to open-loop techniques.

Jung and Pedram [2006] proposed a stochastic dynamic thermal management technique which takes into account the stochastic nature of temperature variation. This technique utilizes DVFS for thermal management. They modeled thermal states and used a Markov decision process to determine the next state. A dynamic thermal manager decides the next action (the thermal state). Based on the temperature, the dynamic thermal manager calculates the cost of each state and selects the next state which has the minimum cost. As a result, this technique guarantees the thermal safety and shows better

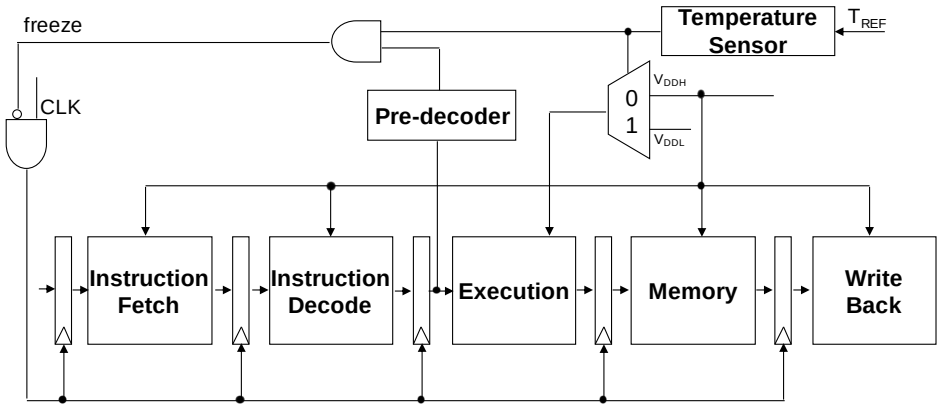


Fig. 5 O^2C Pipeline Architecture [Ghosh et al. 2008]

performance compared to three fixed voltage/frequency settings (1.95V/500MHz, 1.80V/350MHz, 1.65V/200MHz). Shin et al. [2009] instead proposed a DTM technique which considers fan speed. In most high performance microprocessors, there is a cooling fan helping the microprocessor quickly convect heat to the ambient. Conventional thermal management techniques maintain the lowest cooling fan speed just to avoid thermal emergencies. However, those techniques may incur DTM inefficiencies due to temperature-dependent leakage power consumptions. Using the lowest cooling fan speed just to avoid thermal emergencies leads to excessively high temperatures (though under the threshold temperature) in the microprocessor, and substantial temperature-dependent leakage power is consumed. Their proposed technique optimizes energy consumption by using a convex function considering temperature-dependent leakage power and cooling fan power. Their technique reduces energy by up to 17.6%, which means that only considering thermal emergencies may incur energy inefficiencies due to the temperature-dependent leakage power consumption.

In order to target thermal control to specific functional units, Patel et al. [2007] proposed a bank-switching technique for the register files. The architectural insight is that there is little performance loss with the smaller register file. Hence, this technique divides the register file into two banks, the primary and the secondary bank. Each bank is activated periodically, thus halving the power density of the register file. This technique reduces the temperature by 3.4ºC with 0.75% of the performance overhead. Another work concentrating on a specific functional unit is the O^2C (occasional two-cycle operation) architecture proposed by Ghosh et al. [2008]. They applied O^2C to the adder and the multiplier. The overall architecture is shown in Fig. 5. When thermal sensors detect the overheating of the microprocessor, V_{DDL} (lower level of the supply voltage) is supplied to the EX pipeline stage instead of V_{DDH} (the nominal V_{DD}). Consequently, the latency of the EX pipeline stage is increased to two cycles (originally one cycle) due to the reduced supply voltage. This technique reduces throughput by 11% and reduces temperature by 6.6% on average.

Donald and Martonosi [2005] proposed a thermal management technique for Simultaneous Multi-Threading (SMT) architectures. Their proposed technique adjusts the instruction fetch policy of conventional SMT processors. When the integer register file or the floating point register file is overheated, their thread selection mechanism chooses the

coolest thread by looking at the profiled register file access frequency. This technique shows 30% performance improvement as well as 44% ED^2 (Energy-$Delay^2$) reduction, on average, compared to the conventional fetch toggling technique. As another work on SMT architectures, Winter and Albonesi [2008] proposed DTM techniques for clustered SMT architectures. The main drawback with conventional DTM techniques for SMT processors is that the performance of cool threads can also be degraded as a side effect of global DTM techniques, because both cool and hot threads share the resources of the SMT processor in a tightly interleaved fashion. They proposed three main policies for DTM: dispatch gating policies, heat spreading policies, and hybrid policies. The dispatch gating policy simply stops dispatching of instructions issued by hot threads when the temperature reaches the DTM trigger temperature. According to the granularity of dispatch gating control, the dispatch gating policy can be classified into three different policies: global, thread, and cluster-dispatch gating policies. The heat spreading policy is to reallocate threads to clusters for thermal balancing across clusters. Since hot threads are assigned to cool clusters and vice versa, it makes temperatures of clusters more balanced. The hybrid policy is a combined version of the heat spreading policy and DVFS. According to their evaluation results, the hybrid policy shows performance slowdown of 2.8% compared to the baseline (without any DTM techniques) while only adopting DVFS incurs 4.3% performance slowdown compared to the baseline.

With the advent of multi-core architectures (also sometimes referred to as chip multiprocessors or CMPs), Powell et al. [2004] proposed thermal-aware thread mapping and migrating techniques in SMT/CMP environments. The goal of their *heat and run* technique is to maximize the throughput of SMT/CMP without overheating. Their technique consists of Heat-and-Run Task Assignment (HRTA) and Heat-and-Run Task Migration (HRTM). HRTA gathers threads from several cores to one SMT processor, which maximizes the throughput of one execution core (heat). Obviously, the temperature of the heated core increases, and then the threads are migrated from the heated core to the idle cores by HRTM (run). Since there is a resource limitation in one core, threads should avoid the resource contention as much as possible. Thus, they suggested that the combination of threads in the same core have different characteristics, such as combining integer and floating point operations, or computation-bound and memory-bound behavior. Since their main focus is to enhance the throughput using a simple technique, the detailed thermal simulation results are not provided. However, throughput is improved by 9% through leveraging HRTA and HRTM, compared to the conventional temperature management techniques such as DVFS or stop-go (stop-go

Table I. Classifications of thermal management techniques for multi-core microprocessors [Donald and Martonosi 2006]

	No Migration		With Migration	
	Stop-go	DVFS	Stop-go	DVFS
Global	Stop-go	Global DVFS	Stop-go+Migration	Global DVFS+Migration
Distributed	Distributed Stop-go	Distributed DVFS	Distributed Stop-go+Migration	Distributed DVFS+Migration

simply pauses the execution of the cores in case of thermal emergency).

In order to optimize performance of multi-core processors under DTM techniques, Donald and Martonosi [2006] compared many thermal management techniques for multi-core architectures. They classified thermal control techniques for multi-core architectures into twelve techniques. Table I shows their classification. Though only eight techniques are shown in Table I, each *With Migration* technique can be divided again into the counter-based and the sensor-based techniques. Thus, the total number of techniques is twelve. The counter-based techniques use performance counters to detect the thermally intensive cores, while the sensor-based techniques use the thermal sensors. The difference between the global and the distributed technique is a control granularity. The global techniques control all of the cores at the same time, while the distributed techniques can control each core separately. Migration techniques move tasks from hot cores to cool cores. Consequently, the hot cores can be cooled down for a while and the cool cores accommodate the thermally intensive tasks. The migration techniques support the heat balancing of each core to prevent a specific core from being overheated. Since the migration technique is orthogonal to stop-go or DVFS techniques, it can be applied simultaneously with stop-go or DVFS techniques. From the perspective of performance, the distributed DVFS technique shows the best performance result. In another approach for high-performance microprocessors, Murali et al. [2008] proposed a dynamic thermal control technique with a design-time profiling support. It consists of two phases; design-time phase and run-time phase. When a chip is manufactured, a look-up table is embedded for the frequency selection. The look-up table is filled up through a convex optimization process and the appropriate frequency is determined according to the initial temperature and performance requirement (target clock frequency). In the run-time phase, the processor chooses the optimal frequency by looking at this table. This technique reduces 60% of task waiting time. It means that the tasks efficiently avoid the thermal emergency while the performance requirements are satisfied. Chantem et al. [2009] also proposed an optimal DVFS control technique to maximize throughput of a microprocessor that has discrete voltage/frequency levels. Of course, temperature of the microprocessor should be maintained under the maximum allowable temperature. By using their formulations, they found that only two voltage/frequency levels (a high and a low level) out of the entire range of voltage/frequency levels are enough for maximizing throughput. Their proposed DVFS control technique also determines the optimal duration that the microprocessor stays in each voltage/frequency level. By adopting this technique, the microprocessor runs at high speed until the temperature gets to the maximum allowable temperature. Once the temperature reaches the maximum allowable temperature, the voltage/frequency level fluctuates between the high and low levels over the entire execution time. For their evaluation, seven discrete voltage/frequency levels are used – 0.462, 0.615, 0.692, 0.769, 0.846, 0.923, and 1.000 (numbers mean the normalized speed). Their proposed technique selects 0.846 and 0.923 as high and low voltage/frequency levels, respectively. As a reference, they also present a naïve policy that uses the lowest (0.462) and highest (1.000) voltage/frequencies. When adopting their proposed optimal DVFS control technique, throughput is improved by 47.7%, compared to the throughput when adopting the naïve policy.

For Multi-Processor System-on-Chips (MPSoC), Coskun et al. [2008b] proposed an online learning algorithm. Their proposed technique formulates a loss function. The loss function takes into account four factors; hotspots, thermal cycles, spatial gradients, and performance. Note that thermal cycles and spatial gradients are used to consider reliability. The thermal cycle means a temporal spike of temperature fluctuations and the spatial gradient means temperature difference between the coolest and the hottest core. While the conventional algorithms typically consider performance and temperature, this algorithm also considers reliability. Using the loss function, their algorithm chooses the appropriate policy which has the least overhead. On average, the algorithm reduces 20% of hotspots, and 60% of thermal cycles. For more optimization, Zanini et al. [2009] proposed a thermal-aware balancing technique using a control theory for MPSoCs. Fig. 6 depicts an overview of the proposed control system. The thermal balancing regulator is the main thermal controller of the system. To meet the performance requirement of MPSoCs, the regulator receives the frequency requirement. The other input is the thermal profile feedback from the MPSoC. The output of the controller is the regulated frequency. The role of the emergency saturation block in the thermal balancing regulator is to avoid thermal emergency. If the maximum temperature of the MPSoC (Max Temp in Fig. 6) is higher than the pre-defined threshold temperature (T_{max} in Fig. 6), the regulated frequency is saturated to avoid thermal emergency. This technique reduces the task waiting delay by 17.7% compared to the best case of the convex optimization technique explained above [Murali et al. 2008]. In addition, the duration during which temperature differences among the cores are over 4ºC is reduced by 27.45% compared to the convex optimization technique by Murali et al. [2008].

For saving energy while considering temperature, Bao et al. [2008] proposed a DVFS technique with design-time support for MPSoCs. They add a temperature analysis process to the design-time analysis to an existing DVFS technique by Andrei et al. [2007] which only considers the energy optimization under *fixed* temperature. However, the thermal profile of MPSoC dynamically changes at run-time. Since leakage energy consumption is deeply related to temperature, the assumed (fixed) temperature may lead

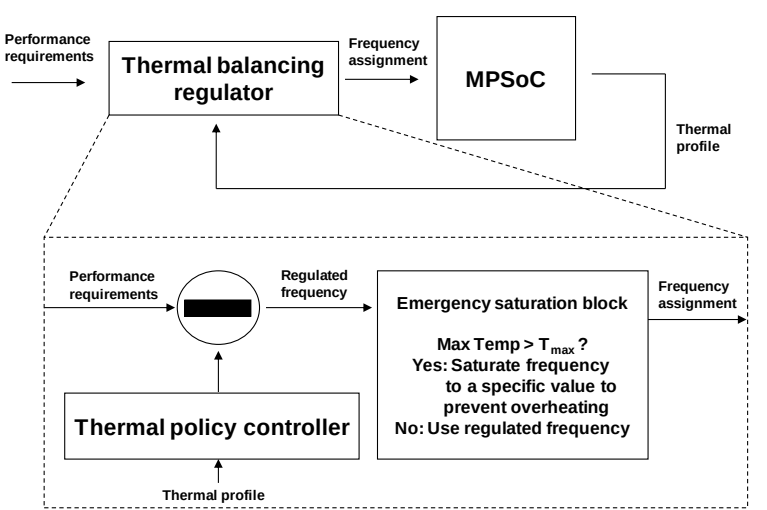


Fig. 6 Overview of the control system proposed by Zanini et al. [2009]

to inaccurate energy estimation at design-time, which eventually results in incorrect voltage and frequency selection for energy minimization. In their proposed technique, to take into account run-time temperature (which is dynamically changed, not fixed) of MPSoCs, they used the HotSpot tool [Skadron et al. 2004]. Based on estimated run-time temperature, they calculate the optimized voltage (both supply voltage and body bias voltage) and frequency to minimize energy consumption considering temperature-dependent leakage. Compared to DVFS using the assumed temperature, the temperature-aware DVFS shows better energy savings (over 4% and 8% at the best case with the MPEG4 and GSM voice codec, respectively).

For many-core (64 cores) designs, Mukherjee and Memik [2006b] proposed a frequency selection technique for multi-core microprocessors. This technique chooses the optimum frequency for the core when a thermal emergency is detected at run-time. The main benefit of this technique is a consideration of neighboring cores. Their proposed technique adjusts the clock frequency of the hot core and its neighboring cores together by applying the DVFS technique to minimize the performance loss. Their proposed technique is fast enough to be applied for run-time thermal management.

Recently, 3D processor technology has emerged and offers many advantages compared to traditional 2D processors: reductions in wire length, chip area, and energy. In 3D microprocessors, however, the increased power density due to stacked dies leads to higher temperature. Accordingly, thermal management is even more important for 3D microprocessors and DTM techniques for 3D multi-core architectures have been developed. 3D-Wave [Sun et al. 2007] was proposed to tackle the temperature problem in 3D MPSoCs. As a basic assumption, their main target is 3D MPSoCs which have two layers, each of which has four cores (the total number of cores is eight). Their algorithm (3D-Wave) tries to balance the power consumption of all the cores, which eventually lowers the peak temperature. Moreover, the supply voltage is scaled to the optimal point that minimizes power consumption and meets the deadline of the task to further reduce the power consumption (PBMCA: for Power Balancing and Minimization – a Constructive Algorithm). When hot tasks are detected, they are migrated by an Iterative Hotspot Migration algorithm (IHM), which iteratively finds the optimum migration decision considering the physical location of the cores. Compared to only PBMCA, 3D-Wave (PBMCA+IHM) shows average temperature reduction of 27.9ºC across benchmark sets including SPEC2000, MediaBench, and AlpBench. Compared to IHM-P (an iterative hotspot migration technique which only considers the power consumption, not temperature), the proposed technique also shows temperature reduction of 6.5ºC.

Several techniques were also proposed to resolve thermal problems in 3D Chip Multi-Processors (CMP). Zhu et al. [2008] proposed a thermal management technique for 3D CMPs by collaboration between the hardware and operating system (OS). Their main goal is to extract the maximum instruction throughput from 3D CMPs. First, to maximize throughput of a 3D CMP, OS determines power-thermal budgets (a voltage/frequency setting) of the cores by considering the physical location of the cores and monitoring temperature as well as activity of the cores. Considering the workload characteristics (IPC: Instruction Per Cycle), the OS assigns tasks to cores. Tasks can also be migrated to maximize performance by considering cooling efficiency of the cores and IPC of tasks. This technique also considers run-time transient thermal behaviors that cannot be statically captured by OS. By engaging DVFS (not global, but per-core), this technique

avoids thermal emergency. As a result, the instruction throughput is improved by 29.84% (on average) compared to the simple distributed DVFS technique [Donald and Martonosi 2006].

Coskun et al. [2009] proposed another technique to manage temperature for 3D multi-core processors. They introduced an adaptive temperature-aware job allocation algorithm (Adapt3D) and a hybrid technique which combines Adapt3D with DVFS. Their algorithm balances application loads considering the location of the cores. Using the thermal index which reflects the location of the cores, the cores which can be easily heated up due to their location have less intensive loads than the other cores. In this technique, a special value is maintained for each core to assign tasks to cores. Each value in the core represents the possibility that a task is fetched to the core. Therefore, the scheduler gives higher priority to cores that have higher value. Note that this value of each core is determined using both the thermal index of each core and thermal behaviors. If the thermal index of the core is high (more prone to be a hotspot), the value of this core is decreased faster. Thermal behavior is incorporated by increasing the value in case that the average temperature in the history window of the core is lower than the preferred temperature and decreasing the value in the opposite case. In summary, the thermal index determines the slope of the value change while thermal behavior determines the rise and fall of the value compared to the previous value. Adapt3D shows almost same performance compared to the default system configuration (dynamic load balancing: the default task scheduling policy in modern operating systems) but considerably reduces hotspots. In addition, more hotspots are reduced when Adapt3D is combined with DVFS (additionally 20~40%).

Since temperature is one of the most crucial factors for microprocessor design, many microprocessor vendors already implemented some DTM techniques in their commercial microprocessors. In Intel Pentium 4 microprocessors, there is a thermal management mechanism, named Thermal Monitor 1 (TM1) [Berktold and Tian 2009]. TM1 is known to periodically stop the microprocessor's clock for up to 2 microseconds, reducing the duty cycle of the microprocessor by 50% (and other duty cycles can be programmed). This is somewhat similar to adjusting frequency. Intel's other thermal management mechanism, Thermal Monitor 2 (TM2), is present in the Pentium M and Core 2 lines [Berktold and Tian 2009]. They also implement TM1, but TM2 adjusts voltage as well as frequency. It can be implemented by adjusting Phase-Locked Loop (PLL) circuits and adding a voltage regulator. Typically, TM2 shows better performance (the performance difference of 4% when the temperature is 77ºC) than TM1 under the same temperature [Rotem et al. 2004]. Turbo boost technology [Intel 2008] enhances performance by raising voltage/frequency of specific cores when there is sufficient room in the thermal or power budget (typically because some cores are under-utilized or idle). This technique is applied to Intel Nehalem-based microprocessors such as the Core i5 and i7, but the frequency boost is limited, so that the active core cannot exceed the rated TDP, even if the temperature is not at a dangerous level. In contrast, Sandy Bridge (Intel's next generation microarchitecture) allows more aggressive boosting because it is controlled by directly monitoring temperature [Gwennap 2010]. AMD microprocessors use Cool'n'Quiet [AMD 2005], which regulates voltage/frequency and fan speed to balance power and temperature considering utilization of microprocessors. When the microprocessor needs high performance, the Cool'n'Quiet technology enables the

microprocessor to raise its voltage/frequency and fan speed. By raising the fan speed together with voltage/frequency, temperature can be maintained below the emergency temperature. In the opposite case that the microprocessor does not need high performance, both the voltage/frequency and fan speed are reduced. While Cool'n'Quiet is for desktop/server microprocessors, PowerNow! [AMD 2005] is another version of Cool'n'Quiet for laptop/mobile microprocessors. The only difference from Cool'n'Quiet is that PowerNow! operates under tighter thermal constraints (laptop/mobile microprocessors have quite lower TDP of around 30~40W compared to the desktop processors' TDP of around 100W) than Cool'n'Quiet. Power7 [Ware et al. 2010] also employs DVFS and the turbo mode, which are similar to Intel's TM2 and Turbo boost, respectively.

3.2.1.2. For Media (streaming) Applications

Unlike general applications, multimedia applications have a streaming feature. Thus, multimedia applications require different thermal management solutions. Srinivasan and Adve [2003] proposed a predictive DTM technique for multimedia applications. Unlike conventional thermal management techniques that are reactive rather than predictive (reactive thermal management techniques engage the DTM operations when the temperature of the microprocessor reaches the threshold temperature), their algorithm takes advantage of the frame rate, which imposes an upper bound on the required performance. Predictability comes from the repeatability of multimedia operations; frame types tend to exhibit similar properties. Their DTM technique using this predictability improves the performance of the processor up to 3.6 times. Lee et al. [2006; 2008] also proposed a GOP (Group of Pictures)-level temperature management technique for MPEG2 decoding. The big difference compared to the previous techniques is that this algorithm avoids thermal emergency by slightly degrading the quality of the frames. The frame quality degrading is divided into the spatial and temporal degradation. The spatial degradation is carried out by omitting the SNR (Signal-to-Noise Ratio) scalability step and the saturation control step among the entire decoding steps. The temporal degradation drops discardable B frames when deadline misses occur. This technique maintains the thermal safety with 0.12 RMSE (Root Mean Square Error) due to the frame quality degradation and a frame-drop probability of 12.5%, on average.

A thermal management technique focused on MPEG4, rather than MPEG2, was proposed by Yeo et al. [2007]. They utilize GOP history to determine the optimum frequency for decoding the current GOP. Generally, MPEG video frames tend to have a similar complexity between the previous GOP and the current GOP since continuous GOPs typically have similar scenes. Thus, we can refer to the previous GOP information for predicting the complexity of the subsequent GOP. The determined frequency is thermally safe as well as satisfying Quality of Service (QoS). To support it efficiently, they improved a feedback controller already proposed by Lu et al. [2003]. The main problem of Lu et al.'s technique is that they did not consider frame complexity: they assume all the frames in the frame buffer have the same complexity. On the other hand, Yeo et al.'s improved feedback controller considers the complexity of frames by referring to the information of successive GOPs. Due to the predictability of GOP information, their DTM technique totally removes frame misses in twelve real-world MPEG4 movies. The proposed technique decreases temperature by up to 13% compared to the

conventional DTM techniques.

For multimedia applications such as MPEG or H.264/AVC, a Hybrid DTM technique using statistical methods for multimedia applications (HDTM) was proposed [Yeo and Kim 2008] to avoid thermal emergency while maintaining QoS. The performance requirement of each application is different among multimedia applications such as MPEG or H.264/AVC. To tackle this problem, the HDTM hardware consists of three main parts, an application characteristics profiler, a thermal characteristics predictor, and an optimal frequency adapter. The application characteristics profiler estimates required clock cycles for decoding frames. Using the estimated clock cycle requirement, they build a Probability Density Function (PDF) of the performance requirement to determine the optimal frequency for each multimedia application. The thermal characteristics predictor predicts the future temperature using Fourier thermal model. The optimal frequency adapter determines the system-wide frequency including overheads in the entire system environment, such as OS overheads as well as the multimedia application itself (the required clock cycles and the predicted future temperature). The HDTM reduces the temperature by 15ºC with the frame drop rate of 0.2%, on average.

3.2.2. For On-chip Caches

Originally, the on-chip caches have been thought to be relatively cool units, because dynamic power is expected to be spatially distributed across the cache structures, and indeed the average temperature of the on-chip caches is relatively low compared to functional units with high localized activity, such as the register files or the ALUs. However, leakage power is increasing and becoming severe, typically accounting for 30~50% of total power dissipation now (though employing sleep-mode transistors can reduce leakage in caches). Since on-chip caches contain so many transistors and occupy such a large area in most microprocessors, their leakage accounts for a large fraction; their high leakage contributes significantly toward their temperature, which in turn exacerbates sub-threshold leakage, which is exponentially dependent on temperature. To reduce excessive leakage power in on-chip caches, many techniques have been proposed (e.g., [Kaxiras et al. 2001; Flautner et al. 2002; Li et al. 2002]). These techniques efficiently reduce cache leakage power while minimizing performance overhead. However, the purpose of these techniques is to mainly reduce leakage power consumed by idle or inactive cache lines or banks; if accessed more often than the sleep mode's timeout period, a line or bank will not be able to sleep. Moreover, many processors are not employing sleep-mode transistors in caches because it has substantial design and validation effort. Thus, if a single cache location is targeted with an intensive access pattern, it may be a hotspot depending on the layout of the cache sub-structures [Sankaranarayanan et al. 2009]. For example, there can be a thermal attack in L1 instruction caches as shown by Kong et al. [2010], and described in Section 3.6.

John et al. [2005] observed thermal behaviors of the on-chip caches in finer granularity and proposed two optimizing techniques; a separated subarray scheme and an interleaved subarray scheme. The separated scheme scatters cache accesses from one to several rows, which reduces the power density of the cache subarrays. In the interleaved subarray scheme (not to be confused with *bitline* interleaving), which is more aggressive than the separated subarray scheme, a subblock pre-decoder allows only one word to be accessed

rather than a full cache block. In summary, only one subarray is accessed at once in the interleaved subarray scheme, while several subarrays should be accessed at once in the separated subarray scheme in order to access a full cache block instead of a word.

Another advance in thermal management for on-chip caches was proposed by Ku et al. [2005]. They proposed three techniques; the first technique is not a thermal-aware but a low-power technique, Selective cache ways with Gated-V_{dd} Architecture (SGA); the second technique is Power density Minimized cache Architecture (PMA); and the last technique is a Block Permutation Scheme (BPS). SGA simply combines the selective cache way technique [Albonesi 1999] with the gated-V_{dd} technique for a low-power cache architecture. Although this technique reduces both the energy consumption and temperature, it does not show significant temperature reduction since that is not its focus. The second, PMA cuts off the V_{dd} signal to the several cache ways depending on the working set of the application. Compared to the SGA which enables V_{dd} signals in the granularity of the entire cache way, the PMA simply distributes enabled cache lines across several cache ways. For example, assuming that there is a 4-way set associative cache and only three ways are used out of the entire four cache ways, in the case of the first row, the PMA disables way3, and enables way0, way1, and way2. In the case of the second row, the PMA disables way2, and enables way0, way1, and way3. The disabled cache way is changed row by row. Consequently, the power density is minimized even though same power is consumed compared to the SGA. The third technique is the BPS. The BPS shuffles the physical location of the cache lines to relieve thermal interaction between adjacent cache lines. This means that the physical location of cache access migrates, distributing heat dissipation, even though contiguous addresses are actually accessed.

3.2.3. Novel Design Techniques

A variety of non-standard microarchitectures have also been shown to alleviate thermal stress. We group these into clustered architectures and other forms of restructuring.

3.2.3.1. Clustered Architectures

A clustered architecture is composed of several *clusters* that are partitioned from the traditional wide-issue superscalar processor. Consequently, one cluster has narrower issue widths than its original processor and has its own copy of the register file (with fewer ports). Clustered architectures were originally studied extensively for their performance benefits in traditional superscalar, out-of-order processors, where the issue queue and the rename register file are large, heavily multi-ported, and hence slow and power hungry. The fact that clustered architectures are smaller also allows higher clock frequency, thus enabling better performance. Researchers also tried to adopt clustered architectures to reduce temperature. Clustered organizations replace centralized, highly multi-ported structures with distributed copies that require fewer ports. This spreads out activity more uniformly across the chip area and reduces the power density in major structures such as the register file. However, some performance overhead is introduced due to communication overheads. Even with dependency-aware steering that tries to group related instructions, Chaparro et al. [2004] report that, compared to a centralized architecture with equivalent issue and in-flight instruction capacity, a 4-way clustered architecture reduces peak temperature by 27% and average temperature by 20% but also

performance by 20%. They also proposed temperature-aware steering (giving priority to cooler clusters) and put one or more clusters to sleep (to allow it to cool off). However, these further increase the performance loss, with only proportional reductions in peak and average temperatures.

Chaparro et al. [2005] also observed that front-end structures can also be power hungry and hot, especially the register-rename stage, which is a complex, multi-ported structure. Renaming and reorder buffers can be partitioned so that each front end renames independently. For example, each rename unit can only refer to physical registers in its corresponding cluster. This reduces the number of ports for each structure, so benefits stem not just from spreading out activity, but also from reducing peak power density. In their proposed scheme, however, reorder buffers are also partitioned with the front ends, complicating commit. In a quad-clustered organization with two front ends (each responsible for two back-end clusters), distributed renaming/reorder buffers/commit only imposes an average 2% slowdown while reducing power density and hence peak temperatures by 32~34% in the rename tables and reorder buffers.

The trace cache can also be modified to exploit the banked nature of such an SRAM structure. A simple technique is bank hopping, in which one or more banks are put to sleep for some time to allow them to cool off [Chaparro et al. 2005]. For small numbers of banks, this requires the introduction of an extra bank; otherwise when one bank is asleep, power density elsewhere would increase. This increases area and may have slight effects on power and speed due to additional interconnect overhead, but leaves the useful trace-cache area unchanged. Another possible technique specific to trace caches is to change the mapping function so that when a new trace is entered into the trace cache (and will presumably exhibit temporal locality and be used again in the near future), it is assigned to the coldest bank [Chaparro et al. 2005]. This requires the introduction of a bank-mapping table for trace-cache reads and writes. These front-end techniques give better thermal results than the back-end techniques of cluster steering or cluster hopping, with temperature savings that are more beneficial than the associated slowdown due to the reduction of leakage energy consumption (recall that there is an exponential dependence of leakage on temperature). Temperature-aware mapping reduces peak temperature in the trace cache by only 4% with a 2% slowdown, while bank hopping reduces it by 12% with a 3% slowdown, and combining the two techniques reduces peak temperature by 14% with a 4% slowdown.

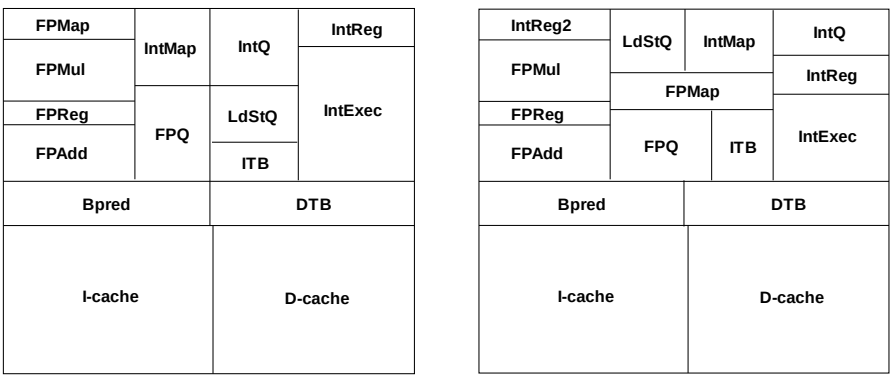
3.2.3.2. Restructured Architectures

In some of the earliest temperature-aware architecture works, Lim et al. [2002] proposed a dual pipeline structure to relieve thermal stress on microprocessors. In the proposed dual pipeline, there is one complex out-of-order pipeline designed for normal operations with a secondary in-order pipeline which is much simpler (less power-consuming) than the out-of-order pipeline. Once the temperature of the microprocessor exceeds the pre-defined threshold temperature, the out-of-order pipeline is clock-gated and the in-order pipeline is used to execute instructions. When the temperature of the microprocessor goes down below the threshold, instructions are executed in the out-of-order pipeline again. The simple in-order pipeline can also be used for mobile devices. For instance, light workloads such as email accesses can proactively be performed in the

in-order pipeline. It reduces energy consumption, which in turn enhances battery life. Their results show 11.4%~12.4% improvement in terms of the energy-execution time metric (energy-delay product) with a small area overhead (4.6%).

Appending an additional pipeline that differs from the others introduces considerable design complexity in microprocessors, even if the added pipeline is a simple in-order pipeline. Instead of duplicating the entire pipeline, duplicating only a specific portion (responsible for hotspots) is more efficient. For more fine-grained pipeline control with simpler hardware organization, an Activity Migration (AM) technique was proposed by Heo et al. [2003]. The AM technique duplicates the execution units (the register file and the ALU) because these units are typical hotspots in microprocessors. The proposed technique also provides several design options. For example, L1 data caches or instruction caches can also be duplicated in addition to the execution unit. At design-time, processor designers determine which functional units are duplicated. Since the original Functional Unit (FU) and the duplicated FU are identical and redundant, the AM technique introduces a significant area overhead and also differs operationally from clustered architectures, where FUs are merely divided into clusters and all the FUs are expected to be available. The AM technique switches the actual units in use at a specified time interval. After an interval, execution switches to the spare unit(s), and the original functional units are idled and placed in a low power state. After one more time interval, the application switches again, and the units in use are continually rotated in this fashion. According to their simulation results, the AM technique reduces the peak temperature by 12ºC with the same voltage and frequency settings.

A similar approach was introduced by Skadron et al. [2003], called *Migrating Computation* (MC). The main difference between the AM and MC technique is that the MC technique may only replicate and migrate the register file, while AM replicates several functional units that are used in one pipeline stage. The other difference is area. Although there is a trade-off between temperature and area, just replicating the register file and ALUs in the AM technique already incurs area overhead by 30%. In contrast, the MC technique concentrates on only the register file and thus incurs much lower overhead. As described by Skadron et al., this simple technique uses a spare unit (Intreg2 in Fig. 7



(a) Original floorplan (b) New floorplan to support MC

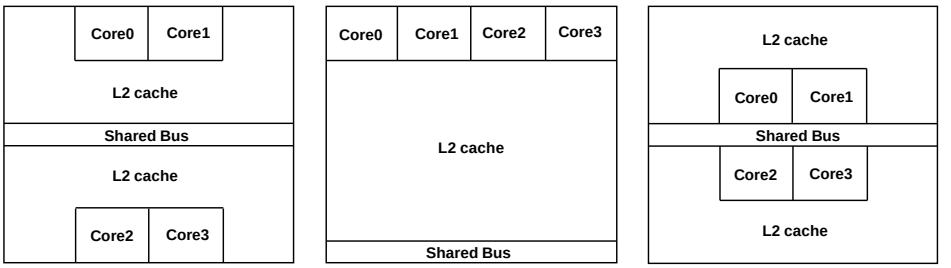
Fig. 7 Alpha 21364 floorplans [Skadron et al. 2003]

(b)) located in cold areas of the chip, to which computation can *migrate* only when the primary unit is overheated. When the primary register file reaches the DTM trigger temperature, the instruction issue is stalled, instructions ready for write-back are allowed to be completed, and then the register file is copied to the secondary register file. Then all integer instructions use the secondary register file, allowing the primary register file to cool down while computation continues unhindered except for the extra computational latency incurred by the greater communication distance. The extra distance is accounted for by charging two extra cycles for every register file access (originally one cycle is consumed to read a value from the original register file). When the primary register file returns below the trigger temperature, the process is reversed and the computation resumes using the primary register file. Fig. 7 depicts the original Alpha 21364 core floorplan (a) and their new Alpha 21364 core floorplan to support the migrating computation technique (b). In Fig. 7 (b), the unit, IntReg2, is introduced to support the MC technique which is located relatively cool area near the floating point units. However, the limitation of the MC is that there is no way to guarantee prevention of thermal violations; overheating is still possible if the hotspot is not directly associated with the register file. Thus, an additional failsafe mechanism is needed, such as DVFS.

Another approach for designing a thermal-aware microprocessor structure is to enlarge typically hot functional units, as proposed by Powell and Vijaykumar [2007]. They proposed a Resource Area Dilation (RAD) technique. Typically hot functional units are enlarged to spread heat well, at the cost of higher latency. S-RAD (Simple-RAD) makes the hot functional unit bigger while the clock frequency of the microprocessor is reduced due to the increased delay of the dilated functional units. Another technique, a P-RAD (Pipelined-RAD) technique, sizes functional units like S-RAD but clock frequency is not reduced, because the dilated functional units are pipelined. By adopting the P-RAD technique, the average throughput is increased by 41% in case of thermally constrained applications because DTM is less frequently triggered than the baseline. However, with thermally un-constrained applications, the RAD techniques show lower throughput because of the increased clock cycle of the dilated functional units or the reduced clock frequency of the microprocessor. Compared to DVFS techniques, P-RAD shows 56% higher throughput with thermally constrained workloads.

Raju et al. [2008] proposed a microprocessor restructuring technique for both performance improvement and die temperature minimization. The proposed technique adjusts the width and height of specific functional units or relocates several functional units. Through the relocation of the functional units, the technique reduces the maximum junction temperature of the Pentium 4 floorplan by 9ºC under the same wire length. By adjusting the width and height of specific functional units on the Alpha architecture, temperature is reduced by 5ºC at the best case.

Microprocessor restructuring has also been studied for multi-core architectures. Li et al. [2006] analyzed trade-offs among processor design parameters for multi-core design under various thermal and area constraints. They assumed three types of thermal constraints: no thermal limit (an upper bound on performance), low constraints (an aggressive cooling solution), and high constraints (a low-cost cooling solution). They evaluated performance in terms of aggregate throughput for a multi-programmed workload as a function of pipeline depth, pipeline width, the number of cores, and L2 cache size. Thermal constraints lead to pipelines with narrower width and shallower



(a) Paired (b) Lined up (c) Centered

Fig. 8 Three types of floorplan in multi-core microprocessor design [Monchiero et al. 2006]

depth: frequency and ILP cost more in thermal throttling than they can provide in performance. More severe thermal constraints lead to fewer cores, because increasing core count incurs too much throttling.

Monchiero et al. [2006] explored the thermal impact of floorplanning and the trade-off among design parameters in multi-core microprocessors. They also found that the wider the issue width, the higher the temperature. In order to analyze the thermal impact of floorplanning in multi-core microprocessor design, they categorized their floorplan into three types. Fig. 8 describes three types of the floorplan. Similar thermal characteristics (the thermal hotspots are cores rather than L2 caches or buses) are observed in each floorplan. With a *paired* and *lined up* floorplan, their maximum temperature is higher by 0.4ºC and 1.2ºC, respectively, compared to the *centered* floorplan. The reason is that the *centered* floorplan can spread heat from the cores to relatively cool areas such as L2 caches or buses, while a *paired* or *lined up* layout cannot spread heat well due to the core's location at the edge or corner of the floorplan. Another factor is thermal coupling between the cores. The *lined up* floorplan places cores directly adjacent to each other, while there are L2 caches or buses between the cores with the *paired* and *centered* floorplans.

In 3D microprocessor design, Puttaswamy and Loh [2007] proposed a thermal-aware microprocessor architecture called *thermal herding*. This technique *herds* microprocessor activities to one die (the nearest die to the heat spreader and heat sink) among four dies. Since the most active die is located near the heat spreader and the heat sink, heat is dissipated more effectively. The utilization of the other dies is restricted. In other words, this technique relieves the thermal problems by reducing the power density and considering the distance to the heat spreader and the heat sink. In order to use only one active die, they used an architectural insight which utilizes narrow-width values; only the lower several bits are significant and the remaining upper bits are filled with '*0*'s. In their assumptions, if a meaningful part of data can be represented within 16 bits (out of a 64-bit word), the data is regarded as a narrow-width value, and vice versa. To support thermal herding in hardware, there is a quarter portion of the datapath in each vertical layer. In other words, each layer is similar to a small microprocessor with a 16-bit data path (originally a 64-bit data path). If operands are narrow-width values, it uses only the die nearest to the heat spreader and heat sink. Although the temperature of the hottest functional unit (L1 data cache) in their 3D thermal-herding architecture is increased by 12ºC compared to the temperature of the hotspot (reservation station) in the 2D planar microprocessor, their 3D thermal-herding architecture reduces the temperature of the L1

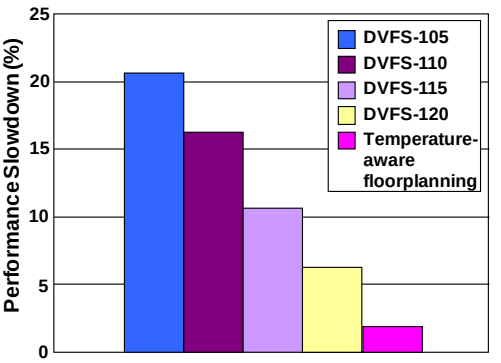


Fig. 9 A performance comparison between the temperature-aware floorplanning and the DTM technique (DVFS with various threshold temperatures). In DVFS-*a*, *a* represents the DTM (DVFS) trigger temperature. [Sankaranarayanan et al. 2005]

data cache by 29% compared to the conventional 3D design's worst-case temperature, regardless of the application.

3.3. Floorplanning

A classic goal of microprocessor floorplanning is performance improvement and energy reduction by decreasing wire length. However, as power density becomes more severe, floorplanning techniques should consider temperature (which itself affects performance) as well as performance improvement and energy reduction. The basis of thermal-aware floorplanning is maximizing the distance of two hot units to prevent thermal conduction while improving the performance and reducing the energy/power consumption. The importance of the temperature-aware floorplanning is shown in Fig. 9 [Sankaranarayanan et al. 2005]. While the performance loss by the DTM techniques is from 6%~21% depending on the threshold temperature, the performance loss with thermal-aware floorplanning is less than 2%. In other words, well-designed floorplans can significantly reduce the performance overheads, which might be caused by the DTM techniques.

3.3.1. For 2D Planar Microprocessors

First, we introduce floorplanning techniques using simulated annealing, the most widely used algorithm. The other techniques using genetic algorithms and linear programming are introduced in the later part of Section 3.3.1.

Sankaranarayanan et al. [2005] developed a temperature-aware floorplanning tool called HotFloorplan. Based on simulated annealing [Wong and Liu 1986] which is a classical floorplanning algorithm, their proposed algorithm considers peak steady-state temperature, as well as chip area and wire delay. Wire length in the critical paths is considered more than that in the non-critical paths. Thus, each wire needs an associated weight. If there are relatively more critical paths between two functional blocks, these two functional blocks are placed adjacently, because their algorithm considers the *weighted* wire length. Their objective function is denoted as follows:

$$Obj = (A + \lambda W)T \quad (1)$$

where A is the chip area, T is the peak *steady-state* temperature for some sets of benchmarks, and W is the aggregate wire-length metric according to $\sum c_{ij}d_{ij}$. c_{ij} is the weighted number of wires connecting two blocks. d_{ij} is the Manhattan distance between the blocks' centers. λ is a heuristic weighting parameter that controls the relative importance of A and W. As a result, this technique reduces the peak temperature by 21.9ºC on average, which leads to performance improvement due to the reduced DTM invocations.

Another simulated annealing based floorplanning algorithm was proposed by Han and Koren [2007]. They added temperature-aware features to the existing floorplanning tool, Parquet [Adya and Markov 2003]. Their algorithm is much faster than Sankaranarayanan et al.'s technique [2005], since it uses an approximation of performance and temperature. The objective function they used is denoted as follows:

$$Obj=C_A*A+C_L*L-C_D*D_T \quad (2)$$

Unlike Parquet's objective function, which only considers the area and the wire length, their proposed algorithm appends the term regarding the thermal diffusion between adjacent blocks. In Equation (2), A represents the area, L represents the wire length, and D_T represents thermal diffusion that is an approximation of the temperature. Note that C_A, C_L, and C_D are coefficients for weighing the terms. For further speedup of the simulated annealing process, this algorithm considers the thermal diffusion of only the top-4 hottest functional units. They also consider the criticality of interconnections using a weighted interconnection matrix, like HotFloorplan [Sankaranarayanan et al. 2005]. The weighted interconnection matrix represents the weighted wire length between functional units considering their interconnection criticality. Using their algorithm, temperature is reduced by 20.6ºC, while total weighted wire length is increased by only 1.7%. This algorithm shows similar temperature reduction (22ºC) compared to HotFloorplan. However, their proposed technique is much faster than HotFloorplan due to the approximation method, though the detailed evaluation results on algorithm running time are not provided.

Although the weighted wire length roughly reflects performance, it does not directly reflect the performance of entire systems. To consider both temperature and performance more accurately, Chu et al. [2007] proposed a thermal-aware floorplanning technique considering CPI. Their objective function is denoted as follows:

$$Obj = W_{area} \cdot \frac{Area}{Area_{norm}} + W_{CPI} \cdot \frac{CPI}{CPI_{norm}} + W_{thermal} \cdot \frac{Thermal}{Thermal_{norm}} \quad (3)$$

As shown in Equation (3), their floorplanning technique considers three factors, area (*Area*), cycles per instruction (*CPI*), and stochastic heat diffusion (*Thermal*). Note that W_{area}, W_{CPI}, and $W_{thermal}$ are weighting factors for each term, and $Area_{norm}$, CPI_{norm}, and $Thermal_{norm}$ are values for normalization. The main problem of the conventional deterministic heat diffusion model is that it is too simplistic to model the temperature accurately. To address this, they proposed a new heat diffusion model, a Stochastic Heat Diffusion Model (SHDM). While the deterministic model only considers the average power density of two adjacent blocks and shared length (the length of the shared edge between two functional units), the SHDM considers the transient behavior of the power density instead of the average power density. Furthermore, the SHDM considers interplay of heat flow between the chip and the ambient or the heat sink temperature. The SHDM

also considers not just adjacency but the degree of the adjacency by using the penetration window, which is a virtual area that represents the degree of a specific block's thermal diffusion effect. Since extremely hot functional units can affect not only adjacent functional units but also faraway functional units, the penetration window improves the accuracy. They showed that their floorplanning technique using SHDM outperforms Sankaranarayanan et al.'s technique [2005] with respect to the algorithm running time (27 times faster in 90nm technology and 19 times faster in 65nm technology). Their proposed technique also reduces CPI by 12.5%, temperature by 3.2ºC, and area by 1.25%, compared to Han and Koren's technique [2007], on average.

Although the simulated annealing algorithm is well suited for floorplanning problems, it has a long running time and lacks scalability as the problem size becomes bigger. Thus, techniques using the other algorithms have been proposed. Hung et al. [2004; 2005] explored temperature-aware floorplanning, but with a focus on IP blocks in a Network-on-Chip (NoC) architecture. They used a genetic algorithm to explore possible mappings to find those that best distribute the steady-state thermal load. Their floorplan reduces temperature by 4~7ºC and incurs less communication traffic, compared to a placement optimized solely for minimum total energy.

Healy et al.'s technique [2007] is based on both the Linear Programming (LP) and the Simulated Annealing (SA) algorithm. Their floorplanning technique consists of two phases. The first step is to specify a width and a height of the functional units and allocate them to the chip using the LP algorithm. In the first step, the floorplanner considers three constraints; 1) no units have overlapping area, 2) after positioning the functional units, the chip should meet performance requirements, and 3) the chip should not incur thermal runaway[2]. After the LP-based floorplanning is finished, the SA-based refinement is carried out (the second step). This technique adopts the SA-based floorplanning step, because only using LP-based floorplanning is not optimal but suboptimal. This suboptimality can be covered by the SA-based refinement. Compared to only using SA-based floorplanning, long running time of SA-based floorplanning can be reduced by using LP-based floorplanning together with SA. Their two step floorplanning technique utilizes advantages of two floorplanning methods (SA and LP). Here is the cost function they used:

$$Cost = \alpha \cdot perf\_wire + \beta \cdot max\_temp + \gamma \cdot area \quad (4)$$

As shown in Equation (4), the SA-based floorplanner considers three factors, wire (*perf_wire*), temperature (*max_temp*), and area (*area*), just like most SA-based thermal-aware floorplanning algorithms. According to their simulation results, the SA-based floorplanner is good for area and wire length, but the temperature is much higher than that from the LP-based or the SA+LP-based (their proposed technique). In contrast, although the LP-based floorplanner is good for the temperature and the algorithm running time, it increases the area and the wire delay. Their proposed SA+LP-based floorplanner shows reasonable results from all perspectives, including temperature, area, wire, and algorithm running time.

3.3.2. For 3D Die-stacked Microprocessors

[2] *Thermal runaway* denotes the case when excessive heat dissipation causes excessive current on transistors, which may eventually incur burn-out of devices or transistors.

As briefly introduced in Section 3.2.1.1, the temperature problem in 3D microprocessors has become more serious than in 2D microprocessors. Since vertical and horizontal thermal conduction should be included in 3D chip floorplanning, floorplanning algorithms for 3D microprocessors are more complicated than those for 2D microprocessors. We first introduce a simulated annealing-based floorplanning technique. The other floorplanning techniques using linear programming, force-directed algorithms, or mixed integer linear programming are introduced later.

To properly manage temperature in the 3D chip floorplanning, Cong et al. [2004] proposed a Combined-Bucket-and-2D-Array (CBA) technique based on the simulated annealing algorithm. Their proposed technique is based on 2D chips but includes bucket structures to represent vertical information of the 3D chips. Temperature information is profiled at every *n*-operation interval, where the value *n* can be specified by chip designers. The cost function takes into account four factors: wire length, chip area, the number of the inter layer vias, and temperature. They proposed three kinds of techniques according to the thermal model used for floorplanning. A combined-bucket-and-2D-array-temperature (CBA-T) is the basic technique that uses the thermal resistive model [Wilkerson et al. 2004]. The CBA-T is accurate but quite slow. To relieve the complexity of the technique, they also proposed a CBA-T-Fast, which is based on a closed-form thermal model[3]. Naturally, this technique is faster but less accurate than the CBA-T. They also proposed the CBA-T-Hybrid which selectively uses the closed-form thermal model and the thermal resistive model. Their evaluation results show that temperature and algorithm running time of CBA-T depend heavily on the value of *n* (temperature profiling interval) while CBA-T-Fast shows the consistent temperature and algorithm running time results regardless of the value *n*. The CBA-T-Hybrid results are between these two points.

Some floorplanning techniques use an algorithm other than simulated annealing. Ekpanyapong et al. [2004] explored temperature-aware floorplanning techniques in 3D chip stacks. Their algorithm uses linear programming, rather than simulated annealing, to search for a solution. Their evaluation compared three types of floorplans: thermal-driven, wire length-driven and profile-driven. The thermal-driven floorplan and the wire length-driven floorplan show almost identical performance and peak temperature results, because their wire length-driven approach concentrates on only reducing the aggregated wire length, which reduces energy and thus temperature. The profile-driven approach mainly considers performance through weighing each wire. As a result, both the thermal-driven and the wire length-driven approach deteriorate performance by about 20~25% compared to the profile-driven approach that purely optimizes the performance, because they sacrifice the performance for their objectives. The thermal-driven approach reduces the maximum temperature by 24%.

Based on 2D floorplanning work (LP+SA based floorplanning), Healy et al. [2007] extended their floorplanning algorithm for 3D. The main consideration is a vertical overlap optimization process whose goal is to compromise among performance, power, and temperature. In order to manage temperature, this technique places frequently communicating functional units closer, while separating thermal hotspots. According to their experimental results, although the area and wire results are not consistent across the

[3] The *closed-form thermal model* considers the vertical and horizontal heat path separately, never considering the interplay between the two heat paths.

benchmarks, their floorplanning technique reduces the temperature by 4~7% compared to the CBA-T [Cong et al. 2004].

Another approach for 3D floorplanning was proposed by Zhou et al. [2007]. They modified the existing force-directed thermal-aware placement technique [Goplen and Sapatnekar 2003; Obermeier and Johannes 2004] for a three-stage 3D floorplanning technique. The first stage is to spread functional units laterally considering temperature. The second stage is to optimize the global placement in 3D spaces. Finally, the functional blocks are assigned to a specific layer. The proposed technique is superior to the thermal-aware CBA [Cong et al. 2004] in all criteria; temperature, area, wire length, the number of vias, and algorithm running time.

In the most recent work, Li et al. [2009] proposed an incremental floorplanning technique using Mixed Integer Linear Programming (MILP), aiming at 3D die-stacked architectures. Their technique applies five methods to the initial floorplan: moving adjacent blocks, moving hotspot blocks, moving the blocks under the hotspot, resizing hotspot blocks, and migrating computation. After the temperature of each block is profiled, the proposed technique computes the potential gain of each modification. The modification which leads to the maximum potential gain among the five modifications is chosen and actually adopted in the floorplan. The iteration is repeated until the optimum point among the temperature, area, and wire length is obtained. Compared to the conventional CBA [Cong et al. 2004], their floorplanning technique further reduces the temperature by 14% and the wire length by 2% with tolerable area (3%) and running time (9%) overhead, on average.

3.4. OS/Compiler Techniques

As thermal problems in microprocessors become severe, many efficient OS and compiler techniques have been proposed for temperature management. OS/compiler thermal management techniques can be more advantageous than hardware-based techniques because they can reduce associated hardware overheads. However, it may need additional data structures to maintain states or information. In this section, we introduce a variety of OS/compiler-directed thermal management techniques.

3.4.1. Thermal-aware Task Scheduling

3.4.1.1. For General Purpose Microprocessors

Conventional task scheduling techniques have focused on performance improvement, without considering temperature. In this section, we explore state-of-the-art thermal-aware task scheduling techniques. Kumar et al. [2006; 2008] proposed a temperature management technique through software-hardware co-operation. The software component (OS) utilizes process/thread priority queues where the processes/threads are waiting in order. In the conventional priority queues, tasks are sorted according to their importance. Contrary to conventional task scheduling, the proposed technique added the thermal-aware features to the conventional priority queues; their policy gives a lower priority to the *hot tasks* (hot task means a task that is likely to exceed pre-defined threshold temperature due to high power consumption according to their thermal estimation model) and vice versa. The operating system monitors thermal behavior of a

task at run-time by looking at the performance counter values and determines whether the task is 'hot' or not. However, without any hardware support, thermal emergency may be inevitable if all tasks are hot tasks. With clock gating as a failsafe mechanism, thermal emergency can be avoided. Their software-hardware combined technique reduces temperature by 4.0~10.5ºC while the performance overhead is 9.9%, on average.

A thermal-aware scheduling technique for Symmetric Multi-Processing (SMP) systems was proposed by Merkel et al. [2005]. The main objective of their scheduling policy is to balance the energy consumptions among many cores, while maintaining a load balancing strategy already adopted in typical operating systems. This technique can be categorized into two main techniques: a passive load balancing and an active hot task migration technique. The passive load balancing technique balances the length of the run queues across the cores by moving tasks between the core queues (run queues). For energy balancing across cores, the OS collects the energy profile of each task. By looking at the profiled energy information of each task, the passive load balancing technique assigns tasks to cores, balancing the energy consumption of each core. However, consider a scenario with an already-running task that is likely to incur the thermal emergency. In this case, the passive load balancing technique cannot do anything since this task is not in the run queue. On the other hand, the active hot task migration can move the task running in a core to the other core in a preemptive manner. Although they did not provide detailed thermal simulation results, the throughput of the microprocessor is increased by 4.7% due to less frequent throttling (less DTM invocations).

Choi et al. [2007] proposed four simple software-driven (OS-driven) techniques: heat-balancing, deferred execution of hot jobs, reducing threading on SMT processors, and cool loop. The heat-balancing technique assigns tasks to the cores in a balanced manner. The second technique, deferred execution of hot jobs, postpones *hot jobs* to be executed later than *cool jobs*. During the execution of the cool jobs, these cores can be cooled down. After the execution of the cool jobs, the execution of the halted hot jobs is resumed. The third and fourth techniques are reducing the multi-threading bandwidth on SMT processors and utilizing the cool loop (which does not do anything, consequently reduces the power consumption) in a single threading environment, respectively. In fact, the cool loop can also reduce the multi-threading bandwidth by running a cool loop in an SMT logical processor[4] among many SMT logical processors in one SMT processor. Their proposed techniques reduce temperature by 3.5ºC on average in a real Power5 based system, while performance overhead is only 1.08%. For multiple clock domain CMPs, another scheduling technique was proposed by Arani [2007]. The task scheduler manages the task queue where the tasks are waiting in order. In this technique, the task scheduler determines the queue length of each core. If the sensed temperature of the core is high, the task scheduler assigns a shorter queue length to the core, and vice versa. If the length of one queue is long, the core which corresponds to this queue is regarded as a performance bottleneck (the temperature of this core will be lowest among the cores). Then, the clock frequency/supply voltage of the other cores (where queue length is short) can be reduced without any performance loss, since it takes a longer time for the bottleneck core to execute the waiting tasks in the queue. This technique balances the

[4] One physical SMT processor is composed of several *SMT logical processors* to execute multiple threads simultaneously. Each logical processor has its own state (e.g., program counter, register file, etc.) to maintain its own context.

temperature of all cores, reducing the overall temperature of the microprocessor.

Another approach for temperature-aware scheduling was proposed by Merkel and Bellosa [2008]. Their technique utilizes a special data structure called a *task activity vector*. The task activity vector for each task contains the values from 0 to 1, which represents the activity degree of each functional unit (the dimension of vectors is the number of the functional units in the microprocessor). Using the task activity vector, their algorithm manages run queues to ensure they do not incur thermal violence. For example, after the integer application is executed, they assign a CPU time slice to the floating point application, cooling down the integer functional units. To take into account multiprocessor systems, their algorithm balances the run queues among processors. Note that fluctuating run queue length across cores means that activities are biased to some specific cores, which can be thermal hotspots. In case of a SMT processor that has several SMT logical processors, if the activities of these SMT logical processors are similar, a specific functional unit is likely to be a hotspot because of the concentrated activities on it. Thus, to spread the activities to various functional units in an SMT processor, activity unbalancing (for instance, a mix of integer-heavy and floating point-heavy threads) is needed among the SMT logical processors. However, to maintain load balanced status across the cores, the quantity of assigned workloads in different SMT logical processors should be similar across SMT logical processors. As a result, the percentage of time when the microprocessor operates over the DTM trigger temperature (80ºC) is reduced from 25% to 6%, leading to less performance penalty.

While many thermal-aware task scheduling techniques have considered spatial correlations between cores or functional units through balancing of workloads, the technique proposed by Yang et al. [2008] considers temporal correlations of thermal behaviors through task scheduling. In other words, their work is focused on choosing the appropriate threads while the temperature of microprocessors is maintained below the DTM threshold temperature. Though the same combination of the applications runs on a core, the execution sequence of the applications significantly affects temperature of the microprocessor. Assuming two tasks (one is a hot task and the other is a cool task) are waiting in a run queue, the hot-cool sequence (the microprocessor runs the hot task first and the cool task later) shows less temperature increase than the cool-hot sequence as shown in Fig. 10. Based on this observation, their scheduling algorithm picks the thread

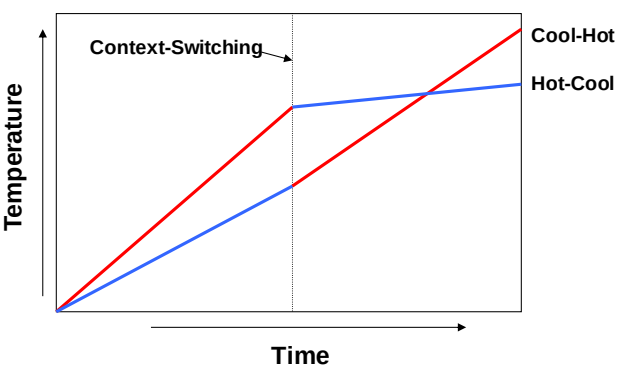


Fig. 10 Thermal impact of the task sequencing [Yang et al. 2008]

which is *the hottest* first, not incurring *the thermal violations*. Consequently, their proposed technique improves the performance by 3.25~4.7% on average with fewer DTM emergencies.

For 3D multi-core microprocessors, Zhou et al. [2008] proposed a thermal-aware scheduling technique called *balancing by stack*. Though balancing heat across the cores has been an effective way to prevent 2D microprocessors from being overheated, it may incur thrashing among the tasks or large fluctuations of the temperature of cores since vertically adjacent cores have strong thermal influences in 3D microprocessors. Fig. 11 depicts their thermal-aware scheduling policy, where the super core is a set of cores which are vertically stacked in the same 2D location but in different layers, and the super task is a set of tasks which are scheduled together. The super tasks are grouped together to have similar power consumption across the super tasks. The task scheduling problem becomes simple as in 2D microprocessors, since the super tasks and the super cores are in 2D space. In other words, the algorithm assigns the super tasks to the super cores while balancing the power consumption across the super cores. In case of thermal emergency, conventional thermal management schemes usually cool down the hottest core. However, their algorithm cools down *the core which consumes the highest power* in the same super core considering the vertical heat convection. Recall that the hottest core is *not always* the highest power consuming core, because the vertical location of the cores is also a crucial factor in 3D microprocessors. Compared to the other algorithms such as the Linux base algorithm, random, round-robin, and balancing by core[5], the algorithm shows much less temperature fluctuation. Compared to the Linux 2.6 scheduler, the proposed technique has 7.22% speedup while the balancing by core technique has only 1.35% speedup.

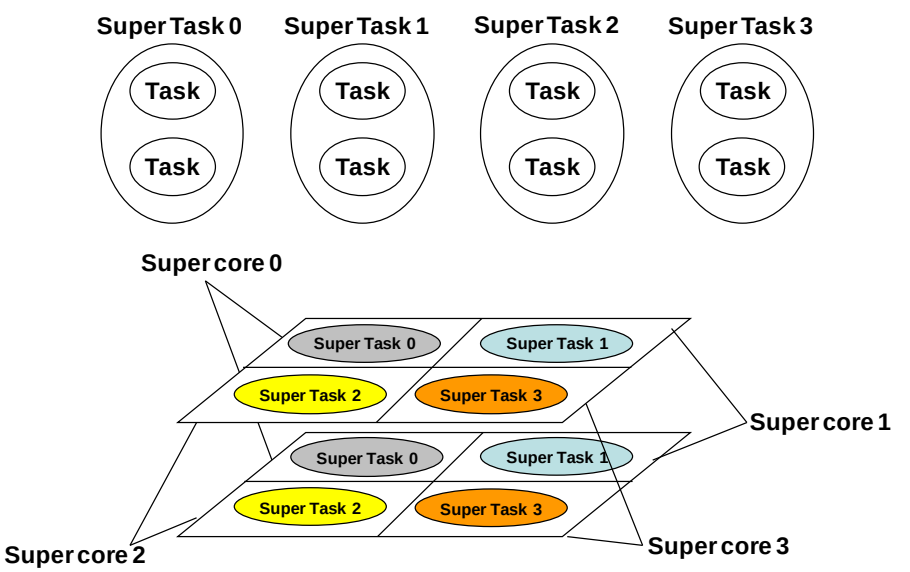


Fig. 11 A thermal-aware scheduling technique in 3D microprocessors [Zhou et al. 2008]

[5] *Balancing by core* schedules the maximum power consuming task to the coolest core, the second highest power consuming task to the second coolest core, and so on.

3.4.1.2. For Embedded/Real-time Microprocessors

There have been several studies for embedded/real-time applications. In real-time applications, tasks should be completed before the predefined deadline. Otherwise, the applications cannot guarantee their QoS. Embedded applications also have a different optimization goal: to minimize energy consumption as much as possible. The temperature optimization process for embedded/real-time applications should therefore be different from that for high-performance processors, because temperature and energy should be optimized jointly, instead of temperature and performance.

Chen et al. [2007] proposed a temperature minimization technique for periodic real-time systems. They extended the technique presented in [Aydin et al. 2001], which is based on Earliest Deadline First (EDF) algorithm. The proposed technique solves the task scheduling problem by using the *n*-approximation algorithm [Vazirani 2001]. For uni-processor systems, they used a 2.719-approximation algorithm to minimize the maximum temperature in the discrete voltage/frequency system. They also extended it for multi-processor systems with the Largest Task First (LTF) policy in a 3.072-approximation bound. Another similar approach was proposed by Yuan and Qu [2007]. Their main aim is to operate the system with minimal energy consumption while the system temperature is kept below the threshold temperature without any deadline miss. They also utilized the EDF algorithm; however, they considered temperature as well as the amount of computation. If the amount of the computation is more than the maximum achievable, the system is simply shut down. Otherwise, the system runs at the lowest voltage level to finish the task before its deadline. This algorithm reduces the system energy consumption by 21% compared to the typical DVFS. Moreover, it alleviates overheating, which leads to less invocation of DVFS so that more tasks can meet their deadline (2% improvement of task completion ratio compared to the typical DVFS).

In order to take into account both soft real-time tasks and best-effort tasks, Jayaseelan and Mitra [2009] proposed a temperature-aware scheduling technique. The proposed technique schedules real-time tasks by looking at their predicted execution time. With the results of the real-time task scheduling, a temperature adjustment phase calculates the starting temperature of the next phase and the timing slack. The temperature adjustment phase ensures the temperature of real-time tasks stays below the threshold. The remaining best-effort tasks are then scheduled in a modified round-robin manner so that the temperature of the microprocessor remains below the threshold. The proposed scheduling policy improves throughput of the microprocessor by 7.4% and 14.4% compared to DVFS and clock gating, respectively.

Several techniques have been proposed to efficiently manage the temperature in multi-core embedded systems. Mulas et al. [2008] proposed a thermal balancing policy for embedded multi-core microprocessors with a task migration technique. Since performance of the microprocessor is sensitive to the size of the migrating task, the migrating task selection considers the overall overhead, including the migration overhead. According to their analysis, as the task size is increased, the number of clock cycles for task migration is also linearly increased. In case that migrating task size is 1024KB, the required delay is about 10 million clock cycles, which are not trivial. Their technique targets the coolest core in determining the destination core. This technique shows much

more balanced temperature results across the cores compared to the stop-go policy or the other energy balancing policies [Merkel et al. 2005]. Note that migration techniques may not be beneficial in case that many parallel workloads occupy all available bandwidth on a chip. Thus, when designing a thermal-aware task migration technique, efficient bandwidth utilization on the chip should be also carefully considered.

Coskun et al. [2008a] proposed a task scheduling technique using integer linear programming for real-time MPSoCs. The proposed technique considers two factors, thermal hotspots and spatial thermal gradients. To minimize thermal hotspots, the proposed technique minimizes the duration of thermal emergency (85ºC in their study). To minimize spatial thermal gradients at the same time, adjacent cores have no assigned work when a task is fetched to the core. It also balances the temperature of each core. Compared to the energy-only minimization technique, the proposed technique reduces the duration of thermal emergency by 35%. Moreover, it reduces 60% of the spatial thermal gradient. They also combined the proposed technique with a coolest-FLP technique [Coskun et al. 2007] for uncertainty of running tasks (called the *hybrid technique* [Coskun et al. 2008a]). Based on the temperature measurement of thermal sensors, the coolest-FLP executes tasks in the coolest core while considering spatial thermal gradients. The hybrid technique shows approximately 5% thermal emergency duration out of the entire execution time, while only coolest-FLP shows over 15% thermal emergency duration. However, as the workload variation becomes severe, the duration of thermal emergency of the hybrid technique converges to that of the coolest-FLP technique.

Chantem et al. [2008] also explored a temperature-aware task scheduling problem for hard real-time applications in MPSoCs. Mixed-Integer Linear Programming (MILP) is used to schedule and assign hard real-time tasks considering both temporal and spatial thermal variations. However, their MILP-based technique has a shortcoming that is not scalable as the problem size becomes large. To overcome this shortcoming, they also proposed a heuristic technique which is based on the binary-searching algorithm. Either steady-state or transient temperature is taken into consideration when the heuristic-based technique searches for a solution. Their MILP-based solution achieves the average temperature reduction of 8.75ºC compared to the energy-optimal technique. Moreover, their proposed heuristic approach (using transient temperature analysis) reduces the algorithm running time by 9.02 times, on average, compared to their MILP-based technique.

3.4.2. Compiler-directed Techniques

Static code analysis by compilers can be applied to thermal management as well as traditional performance optimizations. Since compiler-directed techniques are static, they cannot capture dynamic behavior of programs. However, they do not require hardware support. Since the instruction scheduling of VLIW (Very Long Instruction Word) processors is carried out at compile-time, many compiler-directed temperature management techniques have been proposed for VLIW processors. In VLIW architectures, the thermal behavior of the processor may change, depending on the arrangement of instructions in an instruction bundle and the scheduling of instruction bundles.

Recent compiler-directed thermal management techniques on VLIW processors are

concentrated on both instruction bundle packing and scheduling. Mutyam et al.'s technique [2006] reduces the temperature of the microprocessor through load balancing and IPC tuning. The load balancing is carried out by calculating the predictive dynamic power in advance when the source code is compiled. The compiler assigns the functional units where the calculated dynamic power is distributed most evenly. In order to reduce leakage power consumption, the IPC tuning turns off unused functional units if only a fixed number are active in a loop. For instance, if a loop uses only two out of six integer ALUs, they gate the power supply of the remaining four ALUs, which reduces their leakage. Although IPC tuning reduces the leakage power of the idle functional units, it raises the problem that dynamic power consumption becomes concentrated on the active functional units. Over many loop iterations, this can lead to a hotspot. The authors therefore incorporated a *rotation technique* into the IPC tuning, rotating the active units. Consequently, the temperature of all functional units converges to the average.

Schafer and Kim [2007] introduced another instruction assignment algorithm in VLIW processors. Their target is a Digital Signal Processor (DSP) architecture, for which VLIW organizations are common. They proposed a temperature-aware instruction binding technique (TempIB and TempIB-f) and a NOP insertion technique. The TempIB algorithm binds instructions to cool functional units first. It spreads out the power density to all functional units. The main problem of TempIB algorithm is the long running time (when compiling the source codes) caused by the required thermal simulation that guides instruction binding. The TempIB-f reduces the running time of the TempIB using a simple heuristic while maintaining the accuracy of the thermal simulation. The third algorithm, named TempNOP, inserts NOP instructions when a severe thermal stress is predicted by the thermal simulation during the compilation of source codes. Because the NOP instruction does nothing after it is decoded, it lets the processor rest for a cycle. The number of inserted NOP instructions depends on the degree of the thermal severity.

In addition to VLIW processors, some compiler-directed thermal management techniques are applied to superscalar architectures in which instructions are scheduled dynamically. Hsu and Kremer [2003] proposed a compiler optimization technique for DVFS-based DTM that can be applied to superscalar architectures. In their technique, the compiler generates a table for DVFS invocation when the source codes are compiled. To minimize the performance loss, this technique looks for program regions that are performance-insensitive at compile-time. In these performance-insensitive regions, the voltage and frequency are reduced to minimize energy consumption, which in turn lowers temperature. In their implementation, they extended SUIF2 [Aigner et al. 1999] for their compiler-directed DVFS. This technique reduces energy consumption by 9% and the energy-delay product by 11%.

Narayanan et al. [2006] proposed power density-aware compiler techniques for many-core NoC designs. Their proposed technique considers both performance and temperature when mapping threads to the processor cores. Their algorithm consists of two phases. The first phase reduces the overall power density of the chip. Available threads are mapped in a distributed fashion to reduce the power density of the chip. The second phase divides high-power-density threads into several low power density threads and maps these threads to the cores. Although this technique uses more cores to reduce the power density, it significantly improves performance due to a reduction of DTM triggers.

3.5. Liquid Cooling Techniques

Most high performance microprocessors rely on air cooling to avoid thermal emergency, because it is simple and inexpensive. However, as thermal problems became more serious, researchers began to look for more efficient cooling methodologies. Water is an emerging coolant for microprocessors since it has a high heat capacity. Since thermal problems are much more severe in 3D microprocessors because of vertical and lateral heat conduction, several researchers adopted liquid cooling techniques for 3D microprocessors.

Koo et al. [2005] proposed an indirect liquid cooling technique in stacked integrated circuits, which utilizes micro-channels between each layer. Brunschwiler et al. [2008] proposed a direct liquid cooling technique for 3D integrated microprocessors. Contrary to the indirect liquid cooling technique, the coolant (water) encompasses dielectric in the direct cooling technique, which means water goes through layers. In their proposed structure, there are a water tank, inlet, and outlet. Water flows between the dies and Through Silicon Vias (TSV). Jang et al. [2009] analyzed the architectural impact of the liquid cooling technique in 3D multi-core processors. They analyzed three factors: temperature, leakage power, and reliability. The liquid cooling technique reduces temperature by 45ºC (at the best case) compared to the conventional air cooling technique. Since leakage power consumption is exponentially dependent on temperature, the liquid cooling technique brings a 12.8% average leakage power reduction. The lifetime reliability of the L1 instruction cache (the thermal hotspot in their simulation results) is therefore improved by 97.9% (at the best case).

3.6. Thermal Reliability/Security

Since excessively high temperatures incur errors, thermal reliability and security issues are also important for microprocessors.

Srinivasan et al. [2004; 2005] proposed a temperature-related reliability model (RAMP). Their reliability model is based on five failure mechanisms: ElectroMigration (EM), Stress Migration (SM), Time-Dependent Dielectric Breakdown (TDDB), Thermal Cycling (TC), and Negative Bias Temperature Instability (NBTI). Their Mean Time-To-Failure (MTTF) models are deeply related to temperature. The following equations are failure mechanisms in their reliability model:

$$MTTF_{EM} \propto (J - J_{crit})^{-n} e^{\frac{E_{aEM}}{kT}} \tag{5}$$

$$MTTF_{SM} \propto \left|T_0 - T\right|^{-n} e^{\frac{E_{aSM}}{kT}} \tag{6}$$

$$MTTF_{TDDB} \propto (\frac{1}{V})^{(a-bT)} e^{\frac{X+\frac{Y}{T}+ZT}{kT}} \tag{7}$$

$$MTTF_{TC} \propto \left(\frac{1}{T - T_{ambient}}\right)^{q} \tag{8}$$

$$MTTF_{NBTI} \propto \left[\left\{\ln\left(\frac{A}{1+2e^{B/kT}}\right) - \ln\left(\frac{A}{1+2e^{B/kT}} - C\right)\right\} \times \frac{T}{e^{-D/kT}}\right]^{\frac{1}{\beta}} \tag{9}$$

As shown in Equation (5)~(9), each failure mechanism includes temperature as a

variable (the detailed descriptions of parameters in Equation (5)~(9) are shown in Table II). The critical point in this model is that the MTTF becomes smaller as the temperature is increased. Another important implication of this model is that the reliability of microprocessors depends on which application is running, since thermal behavior depends on application features. Based on their reliability model, they also proposed a Dynamic Reliability Management (DRM) technique [Srinivasan et al. 2004]. Since each processor has a different target reliability design point, the performance can be dynamically adjusted using the DVFS or an architectural adaptation technique. Based on the target FIT (Failure In Time) value, if there is a margin in terms of reliability, applications can be run with higher performance. Otherwise, performance should be degraded to meet the target reliability design point.

For better performance compared to the traditional DTM techniques, reliability banking (simple dynamic reliability management and profiled-based dynamic reliability management) was proposed [Lu et al. 2005]. Designers should rely on temperature-dependent reliability models to derive the expected lifetime of their circuits. Traditionally, a worst-case temperature is used to evaluate the reliability of the system, often resulting in excessive design margins. In addition, under such pessimistic assumptions, DTM techniques may be engaged unnecessarily and incur performance loss. The main concept of the reliability banking is that the lifetime of the microprocessor is *banked* (deposited) when it operates in low temperatures and wear-out accumulates more slowly than expected. The surplus lifetime is then used when operating at higher temperatures (higher than the nominal lifetime consumption rate). When the remaining banked lifetime drops below a threshold value, the microprocessor should engage some DTM techniques. This approach is called Simple Dynamic Reliability Management (SDRM). Since the lifetime consumption rate of DTM cannot exceed the nominal lifetime consumption rate, performance of SDRM is better than that of DTM. However, SDRM only considers the current lifetime balance and the current temperature. To leverage longer-term characteristics of server workloads, Lu et al. also proposed Profile-based Dynamic Reliability Management (PDRM). This is similar to the SDRM during the low-

Table II. Description of parameters in equation (5)~(9) [Srinivasan et al. 2004; 2005]

Equation	Parameter	Description
Common in (5)~(9)	N	Material dependent constants
	K	Boltzmann's constant
	T	Absolute temperature
(5)	J	The current density in the interconnect
	J_{crit}	The critical current density required for electromigration
	E_{aEM}	Material dependent constants
(6)	T_0	The stress free temperature
	E_{aSM}	Material dependent constants
(7)	V	The voltage
	a, b, X, Y, Z	Fitting parameters
(8)	T	The average temperature
	$T_{ambient}$	The ambient temperature
	Q	The Coffin-Manson exponent
(9)	A, B, C, D, β	Fitting parameters

temperature phase. However, in the high-temperature phase, PDRM calculates a new lifetime consumption rate based on the expected duration of the hot phase (profiled information) and the deposited lifetime. The PDRM shows only 6.5% performance slowdown, while the pessimistic DTM suffers the highest performance slowdown (16%) among the three techniques (DTM, SDRM, and PDRM) under the same configuration.

Tiwari and Torrellas [2008] proposed a temperature-related aging-aware technique for multi-core processors. They proposed an aging-aware scheduling technique to alleviate aging effects. Since the clock frequency for all the cores is determined by the slowest core, the proposed technique schedules relatively cool applications to the slower cores, where the pipeline slack margin is tighter than that of the faster cores, and vice versa. Consequently, aging effects are evenly distributed across the cores. The average temperature of each application is measured by per-core temperature sensors. Their aging-aware scheduling policy slows the aging of the critical path delay from 23% to 14%, compared to the random scheduling policy.

In order to measure the reliability of the microprocessor at design-time, a thermal stressmark design is also important to the early-stage thermal-aware design. Joshi et al. [2008] proposed an automated stressmark generation technique. The stressmark can also be used to evaluate the thermal reliability of microprocessors, since it can be used to intentionally make thermal hotspots. Their stressmark shows much higher temperatures (the maximum is over 30ºC) than SPEC benchmarks or commercial applications. The stressmark can also create extreme temperature difference between two functional units. Since the number of thermal sensors in microprocessors is limited, there is a possibility that un-monitored functional units may be burned out without being detected by thermal sensors. The stressmark detects whether the thermal sensor placement or thermal guard-band setting is safe or not in the early-stage of microprocessor design. They provided several stress pattern examples which show the maximum temperature difference of 61ºC between the issue unit and the Load/Store Queue (LSQ).

Thermal security issues in microprocessors have also been explored by several researchers. Security is to prevent an intentional attack by people who have some malicious intention while reliability is to prevent unintended and incidentally caused malfunctions. Dadvar and Skadron [2005] warned of the possibility of potential security and reliability vulnerabilities related to temperature. In an example of thermal security threats, a malicious code was shown able to induce a Denial-of-Service (DoS) attack in an SMT processor [Hasan et al. 2005]. DoS is achieved when malicious threads incur DTM triggers and degrade performance for legitimate threads that are running at the same time. The malicious threads attack typical hotspots such as the register files. Whereas the required time to heat up the register file is 1.2ms, the cool-down time is 12.5ms, which means a short burst of malicious activity can have a much longer DoS effect. To distinguish malicious threads from normal threads, Hasan et al.'s proposed protection technique counts the access rate of typical hotspots such as the integer register file. Generally, malicious threads have abnormally high access rates to these hotspots. The cooling solution is called *selective sedation*, which only sedates suspected malicious threads, while the performance of the normal threads should not be affected. When the normal threads are running with the malicious threads, the IPC of normal threads is degraded by as much as 88.2%. However, by adopting the selective sedation technique, the IPC of the normal threads (1.24) becomes similar to the IPC without malicious threads (1.28).

Another example of a real security threat was introduced by Kong et al. [2010]. Since thermal sensors are typically focused on well-known hot functional units such as the integer register file, unmonitored functional units may be attacked. In Kong et al.'s study, the L1 instruction caches are targets of thermal attack, since they are typically known as a cool functional unit. To successfully attack the L1 instruction caches: 1) a specific portion of the L1 instruction cache should be frequently accessed, and 2) the other functional units should not incur DTM that would eventually neutralize the thermal attack. In order to frequently access only a specific portion of the L1 instruction cache, the malicious code consists of an infinite loop which does not incur any cache miss or branch misprediction. Moreover, by manipulating the address of instructions, their malicious code accesses only specific parts of the L1 instruction cache. To avoid heating other functional units, the malicious code contains NOP instructions that do nothing. To prevent this attack, they proposed a detection technique which utilizes the access counter of each cache data subarray and tag subarray. When a specific data or tag subarray is severely and continuously accessed, the instruction fetching is stalled until the L1 instruction caches are sufficiently cooled down. In addition to the hardware-based protection technique, a software-based screening technique using a malicious code scanner was proposed to detect the attack. According to their evaluation results, the L1 instruction cache tag subarray becomes a thermal hotspot (110ºC) when executing their malicious code, while the temperatures of the other functional units including typical hotspots are sustained below 100ºC. By adopting their hardware-based protection technique, the temperature of the L1 instruction cache tag subarray is dropped by 26ºC, successfully neutralizing the malicious attack.

4. CONCLUSION

In this paper, we introduced recent thermal management techniques for microprocessors. Through a hierarchical categorization of representative thermal management techniques, we have provided a comprehensive overview of recent thermal-management studies. Temperature is a fundamental design consideration because it is directly related to *availability* of the microprocessor, while power and performance are related to *efficiency*. Moreover, temperature is becoming even more of a constraint, as power density increases due to continuing reductions in feature size, while peak supply voltage scaling has slowed and air cooling appears to have reached practical limits [SIA 2009].

Implementing thermal management techniques in microprocessors can be thought to be a complex process, but this need not be the case. *The main reason is that power reduction also leads to temperature reduction*. As shown in this paper, most power-managing hardware components can also be used for thermal management, since temperature is deeply related to power. Thus, microprocessor designers will generally not need to adopt additional hardware components specifically for thermal management (except of course for static thermal management techniques such as floorplanning and novel design techniques). Since the main difference between power and thermal management is the choice of which policy is used, new thermal control algorithm/logic and thermal sensors are enough. It is also worth noting that thermal management techniques used for commercial microprocessors so far are more simple and intuitive

compared to the techniques proposed from academia (though some techniques proposed from academia have been already implemented in commercial microprocessors). The main reason is likely that temperature management entails associated hardware/software costs (though the cost is small). The other important reason is that thermal simulations at design-time entail inevitable simulation error. For example, Jang et al. [2010] reported that using a fixed ambient temperature incurs temperature simulation error of 31.1ºC (at maximum). Thus, simple techniques that have low overhead are preferred in commercial microprocessors. Of course, this does not mean that the academic proposals are impractical! As thermal problems in microprocessors become more severe in the near future, the more aggressive and innovative idea from academia are likely to be considered for commercial processors.

So far, most thermal management techniques have been confined to a single design layer within the system, such as the physical chip design, the microarchitecture, the compiler, or the cooling solution. These techniques usually operate in isolation and may in fact conflict with each other. In practice, most thermal management techniques that are in different layers easily co-exist. Thus, they can provide a multi-layer failsafe mechanism that makes the microprocessor more robust to thermal threats. Even in case of thermal management techniques in the same layer, some of them are complementary. Algorithms to coordinate these techniques in the most efficient way will also be beneficial. Most temperature-aware design has also failed to coordinate thermal management with energy and power-delivery management. Coordination will exploit synergies among techniques and across design layers, and improve the robustness and long-term impact of thermal research. We hope that in the future, research on thermal management will combine efforts from multiple disciplines.

ACKNOWLEDGMENTS

This survey work was supported in part by a grant from the US NSF under grant number CRI-0551630, a grant from Intel Research, and the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R01-2007-000-20750-0). This survey work was also supported in part by the Ministry of Knowledge Economy (MKE), Korea, under the Information Technology Research Center (ITRC) support program supervised by the National IT Industry Promotion Agency (NIPA) (NIPA-2010-C1090-0803-0006). We would like to thank Peter Brownlee Bakkum for his extensive feedback. Finally, we would also like to thank the anonymous referees for their helpful feedback.

REFERENCES

ADYA, S. N. AND MARKOV, I. L. 2003. Fixed-outline floorplanning: enabling hierarchical design. *IEEE Transactions on VLSI*, Vol. 11, No. 6, 1120-1135.

AIGNER, G., DIWAN, A., HEINE, D. L., LAM, M. S., MOORE, D. L., MURPHY, B. R., AND SAPUNTZAKIS, C. 1999. *An overview of the SUIF2 compiler infrastructure*. Computer Systems Laboratory, Stanford University.

ALBONESI, D. 1999. Selective cache ways: on-demand cache resource allocation. In *Proceedings of International Symposium on Microarchitecture (MICRO '99)*, 248-259.

AMD 2005. *Processor utilization with Microsoft® Windows® Media Center Edition on systems enabled with Cool'n'Quiet ™ and AMD PowerNow! ™ technologies*. Application Note, May 2005.

ANDREI, A., ELES, P., PENG, Z., SCHMITZ, M. T., AND AL-HASHIMI, B. M. 2007. Energy optimization of multiprocessor systems on chip by voltage selection. *IEEE Transactions on VLSI*, Vol. 15, No. 3, 262-275.

ARANI, A. S. 2007. Online thermal-aware scheduling for multiple clock domain CMPs. In *Proceedings of IEEE International SOC Conference (ISOCC '07)*, 137-140.

ARS TECHNICA 2008. *NVIDIA denies rumors of faulty chips, mass GPU failures*. Available at: http://arstechnica.com/hardware/news/2008/07/nvidia-denies-rumors-of-mass-gpu-failures.ars.

AYDIN, H., MELHEM, R., MOSS´E, D., AND MEJ´IA-ALVAREZ, P. 2001. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, 95–105.

BAO, M., ANDREI, A., ELES, P., AND PENG, Z. 2008. Temperature-aware voltage selection for energy optimization. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '08)*, 1083-1086.

BERKTOLD, M. AND TIAN, T. 2009. *CPU monitoring with DTS/PECI*. Intel white paper, September 2009.

BORKAR, S. 1999. Design challenges of technology scaling. *IEEE Micro*, Jul.–Aug, 23-29.

BROOKS, D. AND MARTONOSI, M. 2001. Dynamic thermal management for high-performance microprocessors. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA '01)*.

BRUNSCHWILER, T., MICHEL, B., ROTHUIZEN, H., KLOTER, U., WUNDERLE, B., OPPERMANN, H., AND REICHL, H. 2008. Forced convective interlayer cooling in vertically integrated packages. In *Proceedings of the 11th Intersociety Conf. on Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM '08)*, 1114-1125.

CHANTEM, T., DICK, R. P., AND HU, X. S. 2008. Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '08)*, 288-293.

CHANTEM, T., HU, X. S., AND DICK, R. P. 2009. Online work maximization under a peak temperature constraint. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED' 09)*, 105-110.

CHAPARRO, P., GONZÁLEZ, AND J., GONZÁLEZ, A. 2004. Thermal-aware clustered microarchitectures. In *Proceedings of International Conference on Computer Design (ICCD'04)*, 48-53.

CHAPARRO, P., MAGKLIS, G., GONZÁLEZ, J., AND GONZÁLEZ, A. 2005. Distributing the frontend for temperature reduction. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture (HPCA-11)*.

CHEN, Q., METERELLIYOZ, M., AND ROY, K. 2006. A CMOS thermal sensor and its applications in temperature adaptive design. In *Proceedings of International Symposium on Quality Electronic Design (ISQED '06)*, 243-248.

CHEN, C. -C., LU, W. -F, TSAI, C. -C., AND CHEN, P. 2005. A time-to-digital-converter-based CMOS smart temperature sensor. In *Proceedings of International Symposium on Circuits and Systems (ISCAS '05)*, 560-563.

CHEN, J. -J., HUNG, C. -M., AND KUO, T. -W. 2007. On the minimization of the instantaneous temperature for periodic real-time tasks. In *Proceedings of 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '07)*, 236-248.

CHOI, J., CHER, C., FRANKE, H., HAMAN, H., WEGER, A., AND BOSE, P. 2007. Thermal-aware task scheduling at the system software level. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED' 07)*.

CHU, C. -T., ZHANG, X., HE, L., AND JING, T. T. 2007. Temperature aware microprocessor floorplanning considering application dependent power load. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '07)*, 586-589.

CHUNG, S. W. AND SKADRON, K. 2006a. Using on-chip event counters for high-resolution, real-time temperature measurements. In *Proceedings of the Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM '06)*.

CHUNG, S. W. AND SKADRON, K. 2006b. A novel software solution for localized thermal problems. In *Proceedings of the 4th International Symposium on Parallel and Distributed Processing and Applications (ISPA)*, Springer-Verlag LNCS, 63-74.

CONG, J., WEI, J., AND ZHANG, Y. 2004. A thermal-driven floorplanning algorithm for 3D ICs. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '04)*, 306-313.

COSKUN, A. K., ROSING, T. S., AND WHISNANT, K. 2007. Temperature aware task scheduling in MPSoCs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '07)*.

COSKUN, A. K., ROSING, T. S., WHISNANT, K., AND GROSS, K. C. 2008a. Temperature-aware MPSoC scheduling for reducing hot spots and gradients. In *Proceedings of the 2008 Asia and South Pacific Design Automation Conference (ASP-DAC '08)*, 49-54.

COSKUN, A. K., ROSING, T. S., AND GROSS, K. C. 2008b. Temperature management in multiprocessor SoCs using online learning. In *Proceedings of the Design Automation Conference (DAC '08)*, 890-893.

COSKUN, A. K., ROSING, T. S., ALONSO, D. A., LEBLEBICI, J. AND AYALA, J. 2009. Dynamic thermal management in 3D multicore architectures. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '09)*.

DADVAR, P. AND SKADRON, K. 2005. Potential thermal security risks. In *Proceedings of the IEEE Semiconductor Thermal Measurement, Modeling, and Management Symposium (SEMI-THERM '05)*, 229-234.

DONALD, J., AND MARTONOSI, M. 2005. Leveraging simultaneous multithreading for adaptive thermal control. In *Proceedings of the 2nd TACS Workshop*.

DONALD, D. AND MARTONOSI, M. 2006. Techniques for multicore thermal management: classification and new exploration, In *Proceedings of the 33rd annual international symposium on Computer Architecture (ISCA '06)*, 78-88.

EE TIMES 2008. *The truth about last year's Xbox 360 recall*. Available at: http://www.eetimes.com/electronics-news/4077187/The-truth-about-last-year-s-Xbox-360-recall.

EKPANYAPONG, M., HEALY, M. B., BALLAPURAM, C. S., LIM, S. K., LEE, H. -H. S., AND LOH, G. H. 2004. *Thermal-aware 3D microarchitectural floorplanning*. Technical Report GIT-CERCS-04-37, Georgia Institute of Technology.

FLAUTNER, K., KIM, N. S., MARTIN, S., BLAAUW, D., AND MUDGE, T. 2002. Drowsy caches: simple techniques for reducing leakage power, In *Proceedings of International Symposium on Computer Architecture (ISCA '02)*.

GHOSH, S., CHOI, J. H., NDAI, P., AND ROY, K. 2008. O^2C: occasional two-cycle operations for dynamic thermal management in high performance in-order microprocessors. In *Proceedings of the 2008 International Symposium on Low Power Electronics and Design (ISLPED '08)*, 189-192.

GOPLEN, B. AND SAPATNEKAR, S. 2003. Efficient thermal placement of standard cells in 3D ICs using a force directed approach. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '03)*, 86–89.

GUNTHER, S. H., BINNS, F., CARMEAN, D. M., AND HALL, J. C. 2001. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, Vol. 5, No. 1, February 12.

GWENNAP, L. 2010. Sandy Bridge spans generations. Microprocessor Report (www.MPRonline.com), September 2010.

HAN, Y. AND KOREN, I. 2007. Simulated annealing based temperature aware floorplanning. *The Journal of Low Power Electronics*, Vol. 3, No. 2, 1-15.

HASAN, J., JALOTE, A., VIJAYKUMAR, T. N., AND BRODLEY, C. E. 2005. Heat stroke: power-density-based denial of service in SMT. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA '05)*.

HEALY, M. B., VITTES, M., EKPANYAPONG, M., BALLAPURAM, C. S., LIM, S. K., LEE, H. -H. S., AND LOH, G. H. 2007. Multiobjective microarchitectural floorplanning for 2-D and 3-D ICs. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 1, 38-52.

HEO, S., BARR, K., AND ASANOVI´C, K. 2003. Reducing power density through activity migration. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED '03), 217-222*.

HSU, C. -H. AND KREMER, U. 2003. The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '03)*.

HUANG, M., RENAU, J., YOO, S. -M., AND TORRELLAS, J. 2000. A framework for dynamic energy efficiency and temperature management. In *Proceedings of International Symposium on Microarchitecture (MICRO 2000)*.

HUANG, W., SANKARANARAYANAN, K., SKADRON, K., RIBANDO, R. J., AND STAN, M. R. 2008. Accurate, pre-RTL temperature-aware processor design using a parameterized, geometric thermal model. *IEEE Transactions on Computers*, Vol. 57, No. 9, 1277-1288.

HUNG, W. -L., ADDO-QUAYE, C., THEOCHARIDES, T., XIE, Y., VIJAYKRISHNAN, N., AND IRWIN, M. J. 2004. Thermal-aware IP virtualization and placement for networks-on-chip architecture. In *Proceedings of International Conference on Computer Design (ICCD '04)*, 430-437.

HUNG, W.-L., XIE, Y., VIJAYKRISHNAN, N., ADDO-QUAYE, C., THEOCHARIDES, T., AND IRWIN, M. J. 2005. Thermal-aware floorplanning using genetic algorithms. In *Proceedings of International Symposium on Quality Electronic Design (ISQED '05)*, 634-639.

INTEL 2002. *Intel Pentium 4 processor in the 478-pin package thermal design guidelines*. Design guide, May 2002.

INTEL 2003. *Intel Pentium M processor datasheet*. June 2003.

INTEL 2008. *Intel® Turbo Boost Technology in Intel® Core™ microarchitecture (Nehalem) based processors*, Intel white paper. November 2008.

INTEL 2010. *Intel Core2 duo processor E8000 and E7000 Series, Intel Pentium dual-core processor E6000 and E5000 Series, and Intel Celeron processor E3x00 series thermal and mechanical design guidelines*. April 2010.

ISCI, C. AND M. MARTONOSI, M. 2003. Runtime power monitoring in high-end processors: methodology and empirical data. In *Proceedings of International Symposium on Microarchitecture (MICRO '03)*.

JAFFARI, J. AND ANIS, M. 2008. Statistical thermal profile considering process variations: analysis and applications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 6, 1027-1040.

JANG, H. B., YOON, I., KIM, C. H., SHIN, S., AND CHUNG, S. W. 2009. The impact of liquid cooling on 3D multi-core processors. In *Proceedings of IEEE International Conference on Computer Design (ICCD '09)*, 472-478.

JANG, H. B., CHOI, J., YOON, I., LIM, S. -S., SHIN, S., CHANG, N., AND CHUNG, S. W. 2010. Exploiting application-dependent ambient temperature for accurate architectural simulation**.** In *Proceedings of IEEE International Conference on Computer Design (ICCD '10)*.

JAYASEELAN, R. AND MITRA, T. 2009. Temperature aware scheduling for embedded processors. In *Proceedings of the 22nd International Conference on VLSI Design*, 541-546.

JOHN, J. K., HU, J. S., AND ZIAVRAS, S. G. 2005. Optimizing the thermal behavior of subarrayed data caches. In *Proceedings of International Conference on Computer Design (ICCD '05)*.

JOSHI, A. M., EECKHOUT, L., JOHN, L. K., AND ISEN, C. 2008. Automated microprocessor stressmark generation. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA '08)*, 229-239.

JUNG, H. AND PEDRAM, M. 2006. Stochastic dynamic thermal management: a markovian decision-based approach. In *proceedings of IEEE International Conference on Computer Design (ICCD '06).*

JUNG, H. AND PEDRAM, M. 2008. A stochastic local hot spot alerting technique. In *Proceedings of the 2008 Asia and South Pacific Design Automation Conference (ASP-DAC '08)*, 468-473.

KALMAN, R. E. 1960. A new approach to linear filtering and prediction problem. *Journal of Basic Engineering*, Vol. 82, Series D.

KAXIRAS, S., HU, Z., AND MARTONOSI, M. 2001. Cache decay: exploiting generational behavior to reduce cache leakage power. In *Proceedings of International Symposium on Computer Architecture (ISCA '01)*.

KHAN, O. AND KUNDU, S. 2008. A framework for predictive dynamic temperature management of microprocessor systems. In *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '08)*, 258-263.

KONG, J., JOHN, J. K., CHUNG, E. -Y., HU, J., AND CHUNG, S. W. 2010. On the thermal attack in instruction caches. *IEEE Transactions on Dependable and Secure Computing*, Vol. 7, No. 2, 217-223.

KOO, J., IM, S., JIANG, L., AND GOODSON, K. 2005. Integrated microchannel cooling for three-dimensional electronic circuit architectures. *Journal of Heat Transfer*, Vol. 127, 49-58.

KU, J. C., OZDEMIR, S., MEMIK, G., AND ISMAIL, Y. 2005. Thermal management of on-chip caches through power density minimization. In *Proceedings of International Symposium on Microarchitecture (MICRO '05)*.

KUMAR, A., SHANG, L., PEH, L. -S., AND JHA, N. K. 2006. HybDTM: a coordinated hardware-software approach for dynamic thermal management. In *Proceedings of the 43rd annual Design Automation Conference (DAC '06)*, 548-553.

KUMAR, A., SHANG, L., PEH, L. -S., AND JHA, N. K. 2008. System-level dynamic thermal management for high-performance microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 1, 96-108.

KURSUN, E., AND CHER, C. -Y. 2008. Variation-aware thermal characterization and management of multi-core architectures. In *Proceedings of International Conference on Computer Design (ICCD '08)*, 280-285.

LEE, J. S., SKADRON, K., AND CHUNG, S. W. 2010. Predictive temperature-aware DVFS. *IEEE Transactions on Computers*, Vol. 59, No. 1, 127-133.

LEE, K.-J. AND SKADRON, K. 2005. Using performance counters for runtime temperature sensing in high-performance processors. In *Proceedings of the Workshop on High-Performance, Power-Aware Computing (HP-PAC), in conjunction with the 2005 International Parallel and Distributed Processing Symposium*.

LEE, K. -J., SKADRON, K., AND HUANG, W. 2005. Analytical model for sensor placement on microprocessors. In *Proceedings of International Conference on Computer Design (ICCD '05)*, 24-30.

LEE, W., PATEL, K., AND PEDRAM, M. 2006. Dynamic thermal management for MPEG-2 decoding. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design (ISLPED '06)*, 316-321.

LEE, W., PATEL, K., AND PEDRAM, M. 2008. GOP-level dynamic thermal management in MPEG-2 decoding. *IEEE Transactions on VLSI*, Vol.16, No. 6, 662-672.

LI, L., KADAYIF, I., TSAI, Y. -F., VIJAYKRISHNAN, N., KANDEMIR, M., IRWIN, M. J., AND SIVASUBRAMANIAM, A. 2002. Leakage energy management in cache hierarchies. In *Proceedings of 11th International Conference on Parallel Architectures and Compilation Techniques (PACT '02)*.

LI, X., MA, Y., AND HONG, X. 2009. A novel thermal optimization flow using incremental floorplanning for 3D ICs. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference (ASP-DAC '09)*, 347-352.

LI, Y., LEE, B. C., BROOKS, D., HU, Z., AND SKADRON, K. 2006. CMP design space exploration subject to physical constraints. In *Proceedings of the Twelfth IEEE International Symposium on High Performance Computer Architecture (HPCA '06)*, 15-26.

LIM, C. H, ROBERT DAASCH, W., AND CAI, G. 2002. A thermal-aware superscalar microprocessor. In *Proceedings of International Symposium on Quality Electronic Design (ISQED '02)*, 517-522.

LONG, J., MEMIK, S. O., MEMIK, G., AND MUKHERJEE, R. 2008. Thermal monitoring mechanisms for chip multiprocessors. *ACM Transactions on Architecture and Code Optimization*, Vol. 5, No. 2, Article 9.

LU, Z., LACH, J., STAN, M., AND SKADRON, K. 2003. Reducing multimedia decode power using feedback control. In *Proceedings of International Conference on Computer Design (ICCD '03)*, 489-497.

LU, Z., LACH, J., STAN, M., AND SKADRON, K. 2005. Improved thermal management with reliability banking. *IEEE Micro*, Vol. 25, No. 6, 40-49.

MEMIK, S. O., MUKHERJEE, R., NI, M., AND LONG, J. 2008. Optimizing thermal sensor allocation for microprocessors. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD)*, Vol. 27, No. 3, 516-527.

MERKEL, A., BELLOSA, F., AND WEISSEL, A. 2005. Event-driven thermal management in SMP systems. In *Proceedings of the Second Workshop on Temperature–Aware Computer Systems (TACS '05)*.

MERKEL, A., AND BELLOSA, F. 2008. Task activity vectors: a new metric for temperature-aware scheduling. In *proceedings of Third ACM SIGOPS EuroSys Conference*, 2008.

MESA-MARTINEZ, F. J., ARDESTANI, E. K., AND RENAU, J. 2010. Characterizing processor thermal behavior. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '10)*, 193-204.

MICHAUD, P. AND SAZEIDES, Y. 2007. ATMI: analytical model of temperature in microprocessors. In *proceedings of Third Annual Workshop on Modeling, Benchmarking and Simulation (MoBS '07)*.

MONCHIERO, M., CANAL, R., AND GONZÁ LEZ, A. 2006. Design space exploration for multicore architectures: a power/performance/thermal view, In *Proceedings of the 20th Annual International Conference on Supercomputing (ICS '06)*, 177-186.

MUKHERJEE, R. AND MEMIK, S. O. 2006a. Systematic temperature sensor allocation and placement for microprocessors. In *Proccedings of the Design Automation Conference (DAC '06)*, 542-547.

MUKHERJEE, R. AND MEMIK, S. O. 2006b. Physical aware frequency selection for dynamic thermal management in multi-core systems. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design (ICCAD '06)*, 547-552.

MULAS, F., PITTAU, M., BUTTU, M., CARTA, S., ACQUAVIVA, A., BENINI, L., ATIENZA, D., AND MICHELI, G. D. 2008. Thermal balancing policy for streaming computing on multiprocessor architectures. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '08)*, 734-739.

MURALI, S., MUTAPCIC, A., ATIENZA, D., GUPTA, R., BOYD, S. P., BENINI, L., AND MICHELI, G. D. 2008. Temperature control of high-performance multi-core platforms using convex optimization. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '08)*, 110-115.

MUTYAM, M., LI, F., VIJAYKRISHNAN, N., KANDEMIR, M. T., AND IRWIN, M. J. 2006. Compiler-directed thermal management for VLIW functional units. In *Proceedings of ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '06)*, 163-172.

NARAYANAN, S. H. K., KANDEMIR, M., AND OZTURK, O. 2006. Compiler-directed power density reduction in NoC-based multi-core designs. In *Proceedings of International Symposium on Quality Electronic Design (ISQED '06)*.

NAVEH, A., ROTEM, E., MENDELSON, A., GOCHMAN, S., CHABUKSWAR, R., KRISHNAN, K., AND KUMAR, A. 2006. Power and thermal management in the Intel core duo processor. *Intel Technology Journal*, Vol. 10, No. 2, May 15.

OBERMEIER, B. AND JOHANNES, F. 2004. Temperature aware global placement. In *Proceedings of the 2004 Asia and South Pacific Design Automation Conference (ASP-DAC '04)*, 143-148.

PATEL, K., LEE, W., AND PEDRAM, M. 2007. Active bank switching for temperature control of the register file in a microprocessor. In *proceedings of ACM Great Lakes Symposium on VLSI 2007 (GLSVLSI '07)*, 231-234.

POLLACK, F. 1999. New microarchitecture challenges in the coming generations of CMOS process technologies. *International Symposium on Microarchitecture (MICRO '99) keynote speech*.

POWELL, M. D., GOMAA, M., AND VIJAYKUMAR, T. N. 2004. Heat-and-run: leveraging SMT and CMP to manage power density through the operating system. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '04)*, 260-270.

POWELL, M. D. AND VIJAYKUMAR, T. N. 2007. Resource area dilation to reduce power density in throughput servers. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design (ISLPED '07)* 268-273.

PUTERMAN, M. L. 1994. *Markov decision processes: discrete stochastic dynamic programming*. Wiley Publisher, New York.

PUTTASWAMY, K. AND LOH, G. H. 2007. Thermal herding: microarchitecture techniques for controlling hotspots in high-performance 3D-integrated processors. In *Proceedings of International Symposium on High Performance Computer Architecture (HPCA '07)*, 193-204.

RAJU, U., KAISARE, A., AGONAFER, D., HAJI-SHEIKH, A., CHRYSLER, G., AND MAHAJAN, R. 2008. Multi-objective optimization entailing computer architecture and thermal design for non-uniformly powered microprocessors. In *Proceedings of 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM '08).*

REMARSU, S. AND KUNDU, S. 2009. On process variation tolerant low cost thermal sensor design in 32nm CMOS technology. In *Proceedings of ACM Great Lakes Symposium on VLSI 2009 (GLSVLSI '09)*, 487-492.

ROTEM, E., NAVEH, A.,MOFFIE, M., AND MENDELSON, A. 2004. Analysis of thermal monitor features of the Intel® Pentium® M processor. In *Proceedings of Workshop on Temperature-aware Computer Systems (TACS '04).*

SANKARANARAYANAN, K., VELUSAMY, S., STAN, M. R., AND SKADRON, K. 2005. A case for thermal-aware floorplanning at the microarchitectural level. *The Journal of Instruction-Level Parallelism*, Vol. 7, Oct.

SANKARANARAYANAN, K., HUANG, W., STAN, M. R., HAJ-HARIRI, H., RIBANDO, R. J., AND SKADRON, K. 2009. Granularity of microprocessor thermal management: a technical report. Tech. Report CS-2009-03, Univ. of Virginia Dept. of Computer Science, April 2009.

SCHAFER, B. C. AND KIM, T. 2007. Thermal-aware instruction assignment for VLIW processors. In *Proceedings of 11th Workshop on Interaction between Compilers and Computer Architectures (INTERACT '07)*, 1-7.

SHARIFI, S., LIU, C., AND ROSING, T. S. 2008. Accurate temperature estimation for efficient thermal management. In *Proceedings of International Symposium on Quality Electronic Design (ISQED '08)*, 137-142.

SHIN, D., KIM, J., CHOI, J., CHUNG, S. W., CHUNG, E. -Y., AND CHANG, N. 2009. Energy-optimal dynamic thermal management for green computing. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '09)*.

SIA 2009. *Int'l technology roadmap for semiconductors (ITRS)*. Available at http://www.itrs.netreports.html

SKADRON, K., ABDELZAHER, T., AND STAN, M. R. 2002. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA '02).*

SKADRON, K., STAN, M. R., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2003. Temperature-aware microarchitecture. In *Proceedings of International Symposium on Computer Architecture (ISCA '03).*

SKADRON, K., SANKARANARAYANAN, K., VELUSAMY, S., TARJAN, D., STAN, M. R., AND HUANG, W. 2004. Temperature-aware microarchitecture: modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, Vol. 1, No. 1, 94-125.

SKADRON, K. 2004. Hybrid architectural dynamic thermal management. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, Vol. 1.

SRINIVASAN, J., AND ADVE, S. V. 2003. Predictive dynamic thermal management for multimedia applications. In *Proceedings of International Conference on Supercomputing (ICS'03)*.

SRINIVASAN, J., ADVE, S. V., BOSE, P., AND RIVERS, J. A. 2004. The case for lifetime reliability-aware microprocessors. In *Proceedings of 31st International Symposium on Computer Architecture (ISCA '04)*.

SRINIVASAN, J., ADVE, S. V., BOSE, P., AND RIVERS, J. A. 2005. Exploiting structural duplication for lifetime reliability enhancement. In *Proceedings of the 32nd International Symposium on Computer Architecture (ISCA '05).*

SUN, C., SHANG, L., AND DICK, R. P. 2007. Three-dimensional multiprocessor system-on-chip thermal optimization. In *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '07)*, 117-122.

TIWARI, A. AND TORRELLAS, J. 2008. Facelift: hiding and slowing down aging in multicores. In *Proceedings of International Symposium on Microarchitecture (MICRO '08)*. 129-140.

VAZIRANI, V. V. 2001. *Approximation algorithms*. Springer.

VENKATACHALAM, V. AND FRANZ, M. 2005. Power reduction techniques for microprocessor systems, ACM Computing Surveys (CSUR), Vol. 37 No. 3, 195-237, September 2005.

WARE, M., RAJAMANI, K., FLOYD, M., BROCK, B., RUBIO, J. C., RAWSON, F., AND CARTER, J. B. 2010. Architecting for power management: the IBM® POWER7™ approach. In *Proceedings of International Symposium on High-Performance Computer Architecture (HPCA '10).*

WILKERSON, P., RAMAN, A., AND TUROWSKI, M. 2004. Fast, automated thermal simulation of three-dimensional integrated circuits. In *Proceedings of 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITHERM '04)*.

WINTER, J. A. AND ALBONESI, D. H. 2008. Addressing thermal non-uniformity in SMT workloads, *ACM Transactions on Architecture and Code Optimization (TACO)*, Vol. 5, No. 1, May 2008.

WONG, D. F., AND LIU, D. L. 1986. A new algorithm for floorplan design. In *Proceedings of the Design Automation Conference (DAC '86)*, 101-107.

YANG, J., ZHOU, X., CHROBAK, M., ZHANG, Y., AND JIN, L. 2008. Dynamic thermal management through task scheduling. In *Proccedings of the IEEE International Symposium on Performance Analysis of Systems and software (ISPASS '08)*, 191-201.

YEO, I., LEE, H. K., KIM, E. J., AND YUM, K. H. 2007. Effective dynamic thermal management for MPEG-4 decoding. In *Proceedings of International Conference on Computer Design (ICCD '07)*, 623-628.

YEO, I. AND KIM, E. J. 2008. Hybrid dynamic thermal management based on statistical characteristics of multimedia applications. In *Proceedings of the 2008 International Symposium on Low Power Electronics and Design (ISLPED '08)*, 321-326.

YUAN, L. AND QU, G. 2007. ALT-DVS: dynamic voltage scaling with awareness of leakage and temperature for real-time systems. In *Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, 660-670.

ZANINI, F., ATIENZA, D., AND MICHELI, G. D. 2009. A control theory approach for thermal balancing of MPSoC. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference (ASP-DAC '09)*, 37-42.

ZHANG, Y. AND SRIVASTAVA, A. 2009. Accurate temperature estimation using noisy thermal sensors. In *Proccedings of the Design Automation Conference (DAC '09)*, 472-477.

ZHOU, P., MA, Y., LI, Z., DICK, R. P., SHANG, LI., ZHOU, H., HONG, X., AND ZHOU, Q. 2007. 3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design (ICCAD '07)*, 590-597.

ZHOU, X., XU, Y., DU, Y., ZHANG, Y., AND YANG, J. 2008. Thermal management for 3D processors via task scheduling. In *Proceedings of 37th International Conference on Parallel Processing (ICPP '08)*, 115-122.

ZHU, C., GU, Z., SHANG, L., DICK, R. P., AND JOSEPH, R. 2008. Three-dimensional chip-multiprocessor run-time thermal management. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD)*, Vol. 27, No. 8, 1479-1492.