Chapter 6
Thermal Management in Many Core Systems

Dhireesha Kudithipudi, Qinru Qu, and Ayse K. Coskun

Abstract. High power densities and operating temperatures in multi-processor systems impose a number of undesirable effects, including performance degradation, high operational and cooling costs, and reliability deterioration leading to system failures. Many-core systems bring exciting opportunities in design and system management owing to the ample hardware parallelism, while introducing novel challenges due to their complexity and the highly variant workload that is expected to run on these systems. Efficient thermal monitoring and management, designing thermally-aware architectures, and multi-level parameter optimization can alleviate some of the undesirable thermal effects while maintaining the desired performance and energy levels. In particular, these techniques aid in the evolution of green computing systems. This chapter provides a qualitative discussion on thermal management techniques for many-core systems. We will elucidate the following questions in detail: What are the specific design challenges in monitoring the temperature of large-scale systems? How can we exploit the multi-level optimizations at runtime in response to the dynamic behavior of the processor's workload? How do emerging workloads affect the thermal distribution on many-core systems?

6.1 Introduction

Many-core systems offer the potential to deliver 10 to 100 times the processing capacity compared to the traditional single-core systems. It is observed in early

Dhireesha Kudithipudi
Computer Engineering Department, Rochester Institute of Technology
e-mail: dxkeec@rit.edu

Qinru Qu
Electrical and Computer Engineering Department, Syracuse University
e-mail: qiqiu@syr.edu

Ayse K. Coskun
Electrical and Computer Engineering Department, Boston University
e-mail: acoskun@bu.edu

prototypes that by integrating a number of simpler cores on a single die, this many-core chip technology has several advantages including integration levels. Non-ideal power supply scaling and increased power densities (W/cm^2) are among the challenges that come with scaling.

High power densities give rise to large amounts of heat generation. For instance, even when a single core dissipates 1W of power, the overall power dissipated will be around 100W for a 100 core system. This fact, combined with the areal heterogeneity of power consumption in many-core integrated circuits as well as the relatively slow lateral diffusion of heat in silicon [1,2] yields localized thermal hot spots. As a consequence, the strong correlation between temperature and various chip failure mechanisms severely degrades the reliability and life span of the chip. In fact, a small difference in the operating temperature (i.e., 10 °C - 15 °C) can result in a 2X difference in the lifespan of these devices [3,4]. Borkar estimates that the thermal packaging needs increase the total cost per chip by $1/W, when the chip power is 35-40W [5]. Therefore, it is critical to monitor, manage/control on-chip temperatures in order to maximize device lifetimes and assure computational correctness.

In a many-core platform, chip hot spots are strongly workload-dependent. In order to simultaneously maximize performance and reliability in these devices, tasks and resources should be managed in a thermally-aware manner such that temperatures do not exceed critical threshold values and thermal gradients are minimized. Thus, in general, the goal of thermal management is to maximize system performance (or maintain the desired performance) while minimizing temperature hot spots and gradients. Thermal management in many-core systems can be organized into the following three phases: (1) Thermal monitoring, which handles the placement of thermal sensors and the dynamic monitoring of system thermal profile to provide feedback to a runtime thermal management unit; (2) temperature prediction, which handles modeling and forecasting of the runtime thermal behavior for efficient system design and management; and (3) runtime thermal management, which is a set of techniques to mitigate the harmful effects of temperature under energy and/or performance constraints. This chapter discusses each of these three steps in detail.

6.2 Thermal Monitoring

Technology scaling has enabled the integration of many-cores on a single die. This high power density trend has led to overall elevated on-chip temperatures and localized areas of significantly higher temperatures, referred to as hot spots. It has been reported in [6] that the thermal gradient across chips has reached as high as 50 °C, and this value rises with higher operating frequencies. High temperatures and large thermal variations across the chip introduce a set of reliability hazards that may cause both unexpected circuit functionality and weakened physical parameters of the chip. To mitigate such thermal effects, various dynamic thermal management and optimization techniques have been proposed. For any dynamic thermal management to be efficient, it is extremely critical to have an accurate temperature sensing on the chip (e.g., for each core in a many-core system) and relay this information to the thermal management unit.

Several research groups [2, 7–11] have developed optimization algorithms that result in using a minimum number of sensors while maintaining adequate coverage. These optimization algorithms fall into two main categories: uniform and non-uniform sensor placement.

6.2.1 Uniform Sensor Placement

Uniform sensor placement optimization schemes are intended for use with chips that have an unknown typical thermal pattern. The sensors are placed in a uniform static grid throughout the entire chip with the intention of being able to detect all temperature violations, regardless of where they occur on the chip. As mentioned previously, only a finely-grained grid of sensors is capable of achieving near-perfect accuracy. Due to significant cost restrictions associated with sensor overheads, the granularity of the grid must be bound, limiting the accuracy of this model.

A straight-forward linear interpolation approach is proposed in [2] to account for this restriction and refine the temperature measurements. The interpolation scheme with a 4×4 grid of sensors was shown in [2] to improve upon a static uniform grid of the same size with no interpolation by an average of 1.59°C across the SPEC2000 benchmarks [12] in a single-core processor.

One advantage of implementing a uniform thermal sensor allocation technique is that it does not rely on thermal profiling data. No knowledge of hot spot locations and temperatures needs to be acquired prior to implementing a technique of this type. This characteristic, however, limits the accuracy of the uniform grid model because the distances between the sensor locations and the hot spots cannot be minimized. Without knowledge of common resulting thermal maps, sensors arranged in a uniform grid will not aways be able to detect hot spots as accurately as the same number of sensors located near common hot spots.

6.2.2 Non-uniform Sensor Placement

Non-uniform sensor placement optimization schemes are intended for use where thermal maps from typical chip execution across several applications are available for analysis. These types of techniques take advantage of the known hot spots on the chip to determine the most advantageous locations for sensors to be placed. A naive approach is to place a sensor on each hot spot found through thermal profiling across several applications. Unfortunately, this approach is not practical because a high number of hot spots is very likely to occur, and using a large number of sensors is not practical. Ideally, a minimum number of sensors would be arranged on the chip such to provide coverage of all possible hot spots. It has been shown in [13] that hot spots will not always remain in the same locations on the chip during execution of a single program, and various applications running on the same chip will show hot spots in different regions. Hot spot locations and temperatures are application dependent, and it is unlikely that a solution optimized for a single application will be sufficient for other workloads. One sensor placement configuration must suffice for all hot spots that may arise during the execution of any program.

Several methods that detect thermal violations with a limited number of sensors have been developed based on hot spot locations and temperatures found via thermal profiling. Skadron et al [7] have proposed Equation 6.1 to describe the maximum radius R between a hot spot and a potential thermal sensor location, while capping the error to a degree ΔT. The value ΔT denotes the difference between the maximum and minimum temperature value in the chip. In this equation, K is used to represent the effects of the materials included in the chip. These parameters include the thickness of the processor package-die, heat spreader, and thermal interface material multiplied by thermal resistivity factors specific to each material.

$$R = 0.5 \cdot K \cdot \ln(\frac{T_{max}}{T_{max} - \Delta T}) \tag{6.1}$$

6.2.3 Quality-Threshold Clustering

The algorithm described in [10] incorporates Equation 6.1 with the quality threshold (QT) clustering algorithm commonly used in gene clustering [14]. Treating the hot spots as points that must be clustered, the hot spot groupings and corresponding sensor locations are determined based on the values of T_{max} for all of the hot spots in each respective cluster. QT clustering is an iterative technique that assigns hot spots to clusters based on their physical locations on the chip relative to the other hot spots. The sensor location for each cluster is refined after the addition of a candidate hot spot to be the centroid of the included hot spots, thus obtaining the best possible sensor location for the given set of hot spot data points. The newly added hot spot will be kept in this cluster only if every other hot spot in the cluster is located within the distance R from the cluster center.

The method in [10] resulted in placing 23 sensors with T_{max} = 3°C using the QT clustering technique on an Alpha 21364 processor core and the hot spots produced by the SPEC2000 benchmarks. The 23 sensors sensed the complete thermal profile of the core with an average error of 0.2899°C.

Even though the clustering algorithm proves to be sufficient for monitoring thermal events, it does not incorporate the number of clusters or sensors that are available to use for a specific design. This could be detrimental for several reasons. The algorithm does not end execution until every hot spot is placed in a cluster, creating new clusters where necessary to include hot spots that are located far away from the others. The number of sensors required by the QT clustering algorithm may not be available for use in a practical design. To place fewer sensors using QT clustering, the allotted hot spot to sensor distance value must be increased, which may decrease the accuracy of the entire model's results.

6.2.4 K-Means Clustering

The basic k-means clustering algorithm requires the number of sensors to be placed as an input parameter, k. The hot spots are placed into k different clusters, with a temperature sensor placed at the centroid of each cluster. The cluster assignments are chosen such that the mean squared distance from each hot spot to the nearest cluster center is minimized [15]. First, the k cluster centers are chosen randomly from the set of known hot spot points. Each hot spot is then assigned to a cluster C_j such that Euclidean distance $E(O_j,h_i)$ between the hot spot h_i and this cluster's center O_j is minimized. The equation to determine the Euclidean distance between two points is shown in Equation 6.2. In this equation, (h_{ix},h_{iy}) represents the location of a hot spot h_i and (O_{jx},O_{jy}) represents the location of a cluster center O_j in the (x,y) plane.

$$E(O_j,h_i) = (O_{jx} - h_{ix})^2 + (O_{jy} - h_{iy})^2 \tag{6.2}$$

At the end of each iteration, each cluster center is updated to be the centroid of the locations of all hot spots assigned to that cluster. The Euclidean distances between the hot spots and the cluster centers are then recomputed. If a new minimum distance between a hot spot and a different cluster center is found, the hot spot is reassigned to the corresponding cluster. This process is repeated until no hot spot are reassigned to a different cluster or the total sum of all Euclidean distances does not have a significant increase.

A thermal gradient-aware version of the k-means clustering algorithm has been proposed in [8]. The main goal of this approach is to place the temperature sensors to hot spots that typically have higher temperatures. The clusters are formed in 3-D space using each hot spot's temperature, t, as the third dimension of calculating Euclidean distance, as shown in Equation 6.3.

$$E(O_j,h_i) = (O_{jx} - h_{ix})^2 + (O_{jy} - h_{iy})^2 + (O_{jt} - h_{it})^2 \tag{6.3}$$

The cluster centers are updated in each iteration with consideration of hot spot temperature. The hot spots are weighted in the centroid calculation relative to the magnitude of their temperatures. As in the basic k-means clustering algorithm, the cluster centers and hot spot cluster assignments are iteratively refined until no hot spot has been reassigned to a different cluster or the total sum of all 3-D Euclidean distances does not have a significant increase.

As shown in [8], the thermal gradient-aware k-means clustering resulted in an average error at best of 2.10°C placing 16 sensors over a single core and an average error of 1.63°C using 36 sensors. Using the same two numbers of sensors with identical thermal profiling data, the basic k-means clustering algorithm resulted in an average error of 4.58°C and 3.05°C. This shows 2.48°C and 1.42°C improvements, respectively. All experiments were run using HotSpot configured for the Alpha 21364, and hot spot positions were determined from thermal patterns pertaining to the SPEC2000 benchmarks [12].

Although thermal gradient-aware k-means clustering works well under many conditions, this technique is not always optimal in complex hot spot distribution scenarios and may produce solutions worse than the basic k-means approach. In such cases, hot spots are often sorted into inappropriate clusters due to their common temperature and regardless of their physical locations on the chip. Figures 6.1(a) show k-means clustering results with eight sensors. Although the clustering of hot spots varies slightly for the two methods, the sensor placement locations are almost identical. To make a difference in sensor placement, the temperatures of the hot spots used in thermal gradient-aware k-means calculations can be scaled according to Equation 6.4, where a is a constant specifying the steepness of the temperature gradient.

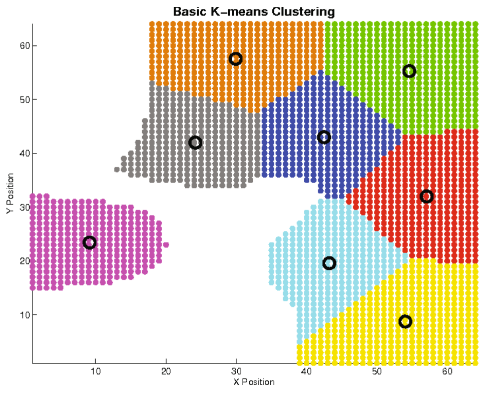
$$NewTemperatures = a \cdot \frac{OriginalTemperatures}{MaximumTemperature} \tag{6.4}$$

Figures 6.1(b) and 6.1(c) show the clustering results of using Equation 6.4 with $a = 1000$ and $a = 2500$, respectively, on the same hot spot set used in Figures 6.1(a). In both situations, the sensors have been placed closer to the hot spots of higher temperature and further from the hot spots of lower temperature. Many of the hot spot cluster assignments, however, are not appropriate spatially. In Figure 6.1(c), for example, many of the red hot spots clustered with sensor S1 would be more appropriately grouped with the gray hot spots clustered with sensor S2, and vice versa.

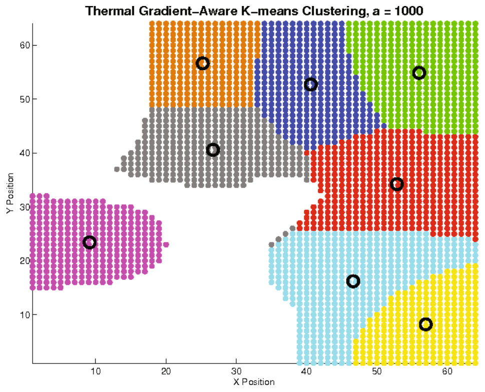
The work in [16] shows that while thermal-gradient aware k-means clustering is effective for single-core processors, it is not appropriate for multi-core processors with strong inter-core thermal interaction due to the unlikelihood of hot spots appearing in the same locations on every core.

6.2.5 Determining Thermal Hot Spots to Aid Sensor Allocation

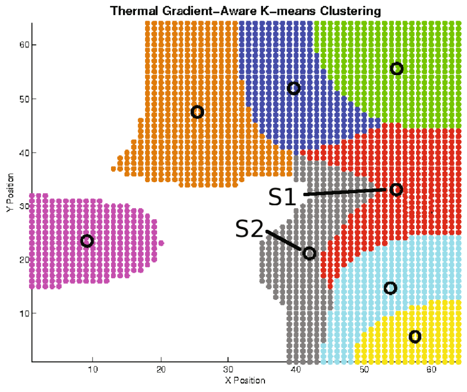
There are many properties to consider when determining which locations on the chip are considered as hot spots. Varying the determination rules has the potential to



(a) Basic K-means



(b) Thermal-Aware K-means, a = 1000



(c) Thermal-Aware K-means, a = 2500

Fig. 6.1. K-means clustering sensor placement variations.

significantly affect the resulting sensor placement locations. The trade-offs between full thermal map characterization and hot spot detection must be considered when identifying initial hot spot locations.

6.2.5.1 Local Hot Spots

One common determination rule is to assign a specified number of hot spots per functional block within the processing core, referred to as *local hot spots* [2, 10]. This technique encourages sensor placement across the entire die and is most appropriate for characterizing the full thermal map of the processor. The hot spot locations will be the points that reach the local maximum temperature for each functional block. For many functional blocks, the hottest points will be the near the edges of the block adjacent to a hotter component.

6.2.5.2 Global Hot Spots

A second method of hot spot determination is to record *global hot spots*, or any location on the die that reaches or surpasses a specified emergency temperature threshold, typically near 82°C or 355 K [2, 16]. Temperature sensors will be placed closer to the locations on the chip of significantly higher temperature, and thus will not likely be spread across the chip. This technique is best for quickly recognizing emergency temperatures as opposed to recovering the full thermal map of the entire processor. Furthermore, the number of hot spots determined by this technique is inversely correlated with the specified emergency temperature threshold. A higher threshold will result in fewer hot spots that could all be located within a single functional block in the processor. Alternatively, a low enough threshold will result in many hot spots, which could potentially cover more than 50% of the processor. A large number of hot spots would allow sensors to be spread through a larger area. As reported in [17], the integer register file is repeatedly the hottest component in the Alpha 21364 core across the SPEC2000 benchmarks. Choosing a high emergency temperature threshold could result in placing sensors only in the integer register file, while a lower threshold would allow sensors to be placed in adjacent functional blocks across the processor. The trade-offs between quick hot spot detection and full thermal map recovery must be considered.

6.2.6 Non-uniform Subsampling of Thermal Maps

In many-core architectures, there is a high likelihood of measuring a very large number of global hot spots. The number of hot spots may be so large that clustering algorithms are not able to place a sufficient number of sensors near the hottest points. To reduce the number of points to be clustered while maintaining clear representation of thermal data, non-uniform subsampling algorithms can be used to obtain a subset of key thermal analysis locations on a chip. More samples are selected from regions of higher temperature and fewer points are selected from regions of lower temperature. Subsampling a thermal map will likely strike a balance between uniform temperature measurement and thermal emergency detection. After the thermal map has been subsampled, clustering algorithms can be used on the subsamples to determine sensor placement locations.

The two gradient based non-uniform subsampling algorithms for images proposed in [18] select sample pixels from a given image such that a constant gradient region will be represented by a number of samples linearly proportional to the gradient magnitude.

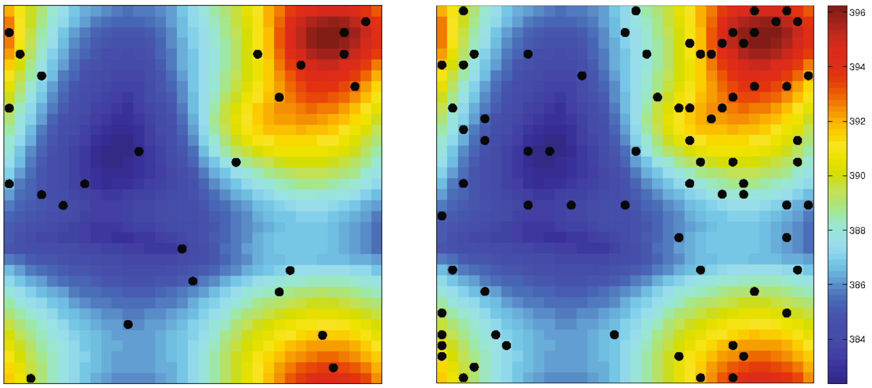
6.2.6.1 Deterministic Subsampling

The deterministic version of this algorithm states that in order to quantize the data set $||\nabla I_1||$ into Q levels, a list of pixel locations I_q with a quantized gradient norm of q must be built for each level q. After all pixels have been distributed into appropriate quantization levels,every s_q^{th} pixel in each list I_q will be selected, where $s_q = \lceil c/q \rceil$ for a constant c. This specification ensures that samples are selected more frequently in regions of high gradient. The constant value c is adjusted to yield a larger or smaller number of samples.

Applying subsampling algorithms to a full thermal map of a processor core while treating the temperature values as gradients results in more samples closer to the regions with more hot spots and fewer samples near cooler regions. Adjusting the value of c affects the number of samples taken from a thermal map. Before performing subsampling, the temperature should be normalized according to Equation 6.5 using the minimum temperature T_{min} and maximum temperature T_{max} in the thermal map used for analysis.

$$I_{norm} = \frac{||\nabla I_1|| - T_{min}}{T_{max} - T_{min}} \tag{6.5}$$

Performing the deterministic non-uniform subsampling algorithm on a sample thermal map yielded the sampled results displayed in Figure 6.2(a) and 6.2(b). Both runs used 25 quantization levels. Figure 6.2(a) shows the results from setting the constant $c = 5$. This constant produced many fewer samples than setting the constant $c = 0.25$ as shown in Figure 6.2(b). Both plots reveal sampling locations spread throughout the entire thermal map. This algorithm is fairly conservative and similar to uniform sampling.



(a) Thermal deterministic subsamples for $c = 5$
(b) Thermal deterministic subsamples for $c = 0.25$

Fig. 6.2. Thermal deterministic non-uniform subsampling results with 25 quantization levels.

6.2.6.2 Stochastic Subsampling

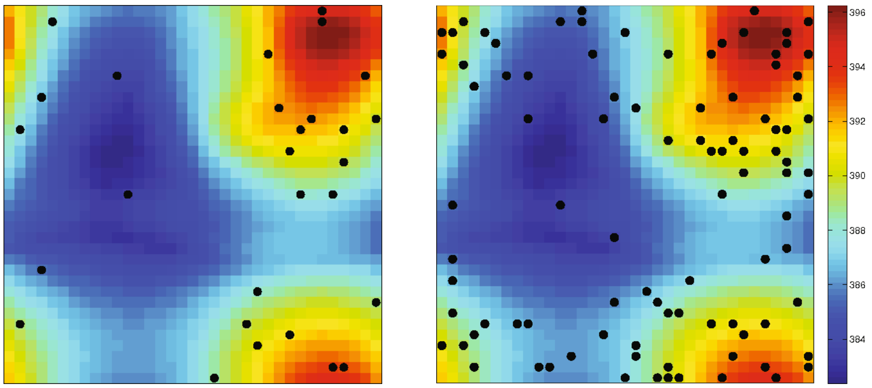
A stochastic version of non-uniform sampling looks at each individual pixel location (i, j) and decides whether to select this pixel as a sample or not. Pixels are selected with probability $p(i, j) = min(\alpha * ||\nabla I_1||(i, j), 1)$. Adjusting the proportionality constant α yields fewer or additional samples.

Performing the stochastic non-uniform subsampling algorithm on a sample thermal map yielded the sampled results displayed in Figure 6.3. Figure 6.3(a) shows the results from setting the constant $\alpha = 1.5$, which produced fewer samples than setting the constant $\alpha = 5$ as shown in Figure 6.3(b). Both plots show many sampling points in the hottest regions, and only one or two samples in the coolest region. The samples taken in this stochastic subsampling algorithm accurately reflect the thermal gradient of the chip.

K-means clustering can be used to determine appropriate sensor locations when treating the samples as points to be clustered. Due to the sampled locations, the resulting thermal sensor placement will be able to maintain a balance between profiling the entire core and detecting thermal emergencies.

6.3 Temperature Modeling and Prediction Techniques

Temperature is a prominent factor in determining the performance, reliability, and leakage power consumption of modern processors. As a result, estimating temperature accurately is an essential step in successful thermal management. This



(a) Thermal stochastic subsamples for $\alpha = 1.5$

(b) Thermal stochastic subsamples for $\alpha = 5$

Fig. 6.3. Thermal stochastic non-uniform subsampling results.

section first discusses thermal modeling principles and tools. The goal of thermal modeling is to provide an accurate assessment of temperature to help system design and to enable evaluation of runtime scenarios. The second part of the section discusses thermal prediction, which relies on low cost techniques to estimate current or future temperature based on thermal sensors readings or power/performance profiles. Thermal predictions can be utilized as early warnings in predictive dynamic thermal management policies (see Section 6.4) or for providing more comprehensive thermal estimates for systems without a sufficient number sensors. In both cases, the main objective of thermal prediction is to improve decision making during thermal management.

6.3.1 Thermal Modeling

There are several temperature measurement methods at chip level. Gathering the temperature data by using point contact methods (thermocouples) is limited by the large number of points to be monitored and the small size of the components. Connecting tens or hundreds of thermocouples is very time consuming. Infrared (IR) thermal imaging (e.g., [19, 20]) is a new technique which addresses these issues by providing comprehensive two-dimensional (2-D) maps of thousands of temperatures in a matter of seconds. This is accomplished without the need to make contact with the components. This approach is, however, expensive and time-consuming and can only be applied post-design. Finally, on-chip thermal sensors provide temperature readings as well; however, the number of sensors that can be deployed on a chip is limited by chip design constraints such as area and cost (efficient sensor placement techniques are reviewed in Section 6.2). It is, therefore, important to have full-chip thermal models and simulation tools that can provide the temperature profile of the die.

On a chip, heat is generated in both the substrate and the interconnections. Additional power dissipation results from Joule heating (or self-heating) caused by the flow of current in the interconnect network [21]. Although interconnect Joule heating constitutes only a small fraction of the total power dissipation in the chip, the temperature rise in the interconnections due to Joule heating can be significant. This is because of the fact that interconnects are typically located away from the heat sink by several layers of insulating materials which have lower thermal conductivities than that of Silicon.

The major source of heat generation is the power dissipation of devices that are embedded in the substrate. The operating temperature of a VLSI chip can be calculated from the following linear equation:

$$T_{chip} = T_a + R_\theta \frac{P_{tot}}{A} \tag{6.6}$$

where T_{chip} is the average chip (Si junction) temperature, T_a is the ambient temperature, P_{tot} (in W) is the total power consumption, A (in cm^2) is the chip area, and R_θ is the equivalent thermal resistance of the substrate (Si) layer plus the package and heat sink ($cm^2 \deg C/W$). Thus, to compute temperature, we need to compute or measure the power consumption, construct the chip thermal model to calculate the thermal resistances, and have information on the environment (i.e., ambient temperature).

Power consumption of a VLSI chip consists of dynamic power, short-circuit power, and static (or leakage) power, out of which dynamic and leakage power are dominant in today's processors. Power modeling and measurement are closely connected areas to thermal modeling. Current power measurement practices involve using current sensors and power meters for chip or server level measurements (e.g., [22,23]), or using voltage and current sensors on the chip, if available. Processor power modeling tools include architecture-level models such as Wattch [24] or McPat [25], which can be connected to instruction-accurate performance simulators, or circuit-level models that are available in commonly used circuit design/analysis tools.

While Equation 6.6 computes the temperature of an entire chip, calculating the temperature of individual blocks (e.g., microarchitectural units or cores) or of fine-grained grid cells requires constructing a thermal model to simulate the heat transfer among the units on the chip as well as the heat flow from the junction to air.

HotSpot [26] provides an automated thermal modeling tool to construct the thermal R-C network given the package properties, chip properties, and the chip layout. The tool is capable of providing both steady state and transient temperature response estimates for a given power consumption trace. Figure 6.4 demonstrates the R-C network for the chip layer. The heat spreader and heat sink are also modeled through a similar network.

To speed up performance and thermal simulation, one method is to emulate the core execution on FPGA platforms and run an automated thermal R-C network based model for computing temperature in conjunction with the performance emulation [28]. Recent extensions of automated thermal models include modeling capabilities for 3D stacked systems [29,30] and for microchannel-based liquid cooling [31].

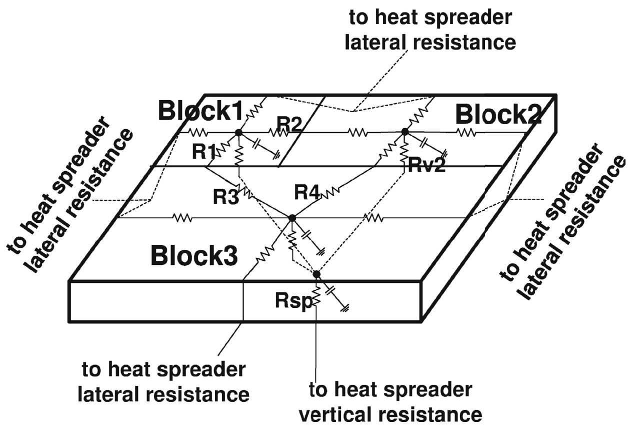

to heat spreader lateral resistance
Block1
R2
Block2
R1
Rv2
R3
R4
Block3
Rsp
to heat spreader lateral resistance
to heat spreader lateral resistance
to heat spreader lateral resistance
to heat spreader vertical resistance

Fig. 6.4. Thermal R-C network for the chip [27]

In addition to these thermal simulation tools specifically targeting estimating chip temperature, general purpose finite element solvers can be utilized for temperature modeling as well.

The automated R-C network based thermal simulators provide high accuracy in modeling [32–34]. These models, however, have considerable runtime overhead, preventing them to be leveraged for runtime temperature estimates. For this reason, several low-cost temperature prediction (or forecasting) techniques have been proposed. Next, we discuss thermal prediction methods.

6.3.2 Temperature Prediction

Computing chip temperature using the linear equation (Eqn. 6.6) is one method for quick temperature estimation, and this method has been used in temperature-aware design optimization [35]. Another method of temperature prediction focuses on extrapolating the temperature values on the chip in detail based on temperature readings from a few sensors. This approach is especially useful for large many-core systems, considering a sufficient number of thermal sensors may not be available (see Section 6.2 for a detailed discussion on thermal sensor placement methods).

Sharifi et al. introduce a technique to accurately estimate the temperature at arbitrary locations on the die based on the noisy temperature readings from a limited number of sensors which are located further away from the locations of interest [36]. Their technique first constructs a R-C network of the chip offline. Model order reduction is used to generate a smaller yet accurate system for the thermal model. After the thermal model is constructed, the technique applies Kalman filtering to the reduced order model of the system. The calibration ends when the filter reaches its steady state. The resulting steady-state filter is used at runtime to perform the temperature estimation. The Kalman filter estimates the temperature in a predict-correct manner based on a few temperature sensor values and power consumption estimates. At runtime, the predictor projects temperatures using the current temperatures. A "measurement update" stage following the projections incorporates the new measurements into the *a priori* estimate to obtain an improved *a posteriori* estimate of the temperature.

Another method for fine-grained temperature estimation using a limited number of thermal sensors is using Fourier analysis techniques [37]. Specifically, the authors use Nyquist-Shannon sampling theory to determine the fewest number of thermal sensors that can fully characterize the runtime thermal status of a processor. Given a limited number of thermal sensors, the technique applies signal reconstruction techniques that generate a full-resolution thermal characterization of a processor. Figure 6.5 demonstrates the flow of the runtime thermal characterization. The technique first takes the Fast Fourier Transform (FFT) of the temperature samples and the space-domain representation of the interpolation functions (sinc, nearest neighbor, cubic B-spline, linear function). It then multiplies the resultant spectral-domain representations to get the FFT of the reconstructed 2D thermal signal. Finally, inverse FFT (IFFT) is applied to get the full-resolution thermal characterization in the space-domain. The authors also propose methods that handle uniform and non-uniform sensor placements [37]; note that non-uniform placements may arise due to design and layout constraints.

Instead of using a few number of sensors for a detailed estimation of the on-chip temperature, some prediction techniques leverage the existing sensors and the thermal measurement history to predict the future temperatures. Detailed thermal estimates are useful in temperature management as the management policies can make more informed decisions using the fine-grained thermal projections. Temperature forecasting, on the other hand, focuses on learning the temperature behavior and estimating the near future. In this way, new policies can be designed that proactively make temperature-aware decisions, such as off-loading a core that is projected to get hotter within the next prediction interval [38] (see Section 6.4 for a detailed discussion of temperature management).

Yeo et al. estimate short-term (application-level) temperature changes using least square regression fit over the thermal sensor data, and compute core-level (slower) temperature changes based on steady-state temperature of the application [39]. The overall temperature estimate is then a weighted sum of short-term and long-term temperature estimates, where the weight factors are selected empirically. Prediction experiments on SPEC 2006 suite demonstrate high accuracy.

While application-based prediction through least squares fit provides good estimates when the predictor is trained with the application thermal characteristics, there are several challenges associated with the method. First, when the prediction distance (i.e., how many steps or seconds into the future are being predicted) is altered, the prediction accuracy may vary significantly [38]. The reason is that, as soon as we predict more than a few time steps into the future, the term with the biggest exponent in the least-squares fitting function dominates, and the prediction accuracy degrades from that point on. Another challenge is runtime adaptation: for dynamically

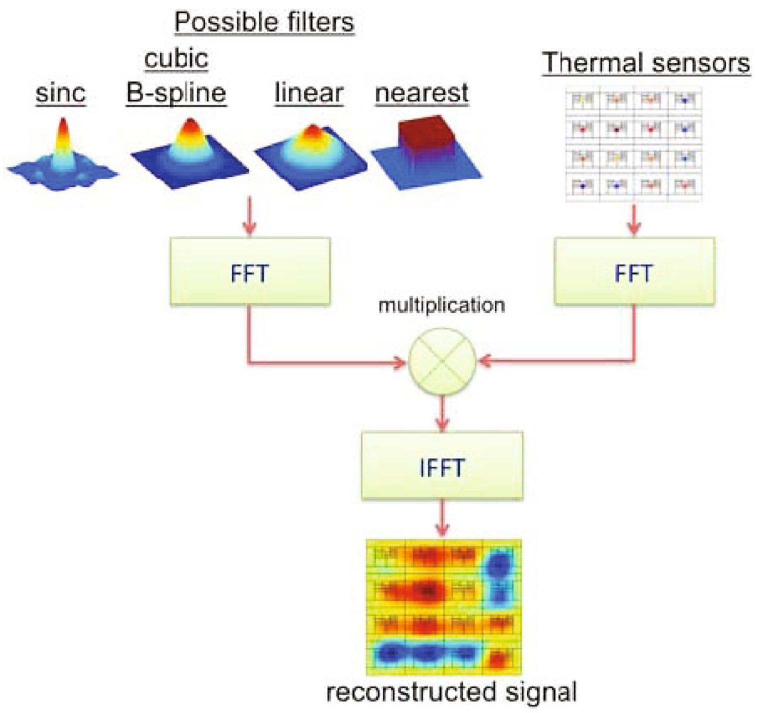


Fig. 6.5. Flow of the thermal characterization technique using FFT-based analysis [37]

changing workload, recursive least-squares continuously needs to update the coefficients of the model as new data arrive (otherwise, accuracy would drop).

Auto-regressive moving average (ARMA) modeling for forecasting temperature [38, 40] addresses the challenges outlined above through construction a predictor using the following steps. *(1) Identification of the model degree and estimation of model parameters.* A well-fitting ARMA model has a low final prediction error [38]. *(2) Checking the model.* The error between measured and predicted values should be randomly distributed around a mean value of zero. *(3) Online adaptation using sequential probability ratio test (SPRT).* The probabilistic test rapidly determines if the existing model is not fitting the current workload temperature dynamics and if a new model should be constructed.

$$y_t + \sum_{i=1}^{p}(a_i\, y_{t-i}) = e_t + \sum_{i=1}^{q}(c_i\, e_{t-i}) \tag{6.7}$$

An ARMA(p,q) model is described by Equation 6.7. In the equation, y_t is the value of the series at time t (i.e., temperature at time t), a_i is the lag-i autoregressive coefficient, c_i is the moving average coefficient and e_t is called the noise, error or the residual. The residuals are assumed to be random in time (i.e., not autocorrelated), and normally distributed. p and q represent the orders of the autoregressive (AR) and the moving average (MA) parts of the model, respectively.

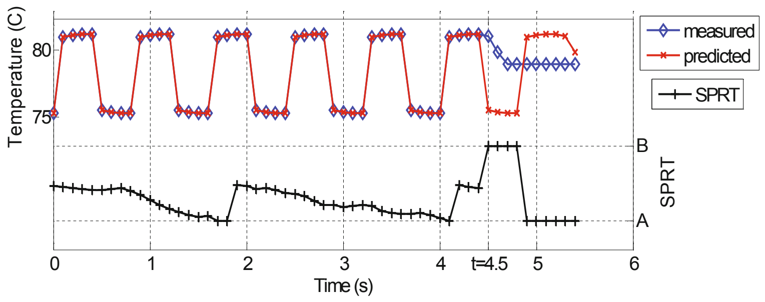


Fig. 6.6. ARMA-based temperature prediction and online detection of variations in thermal characteristics using a likelihood ratio test [38]

Figure 6.6 demonstrates the predicted and estimated temperature values using ARMA. Note that during this experiment, temperature dynamics change, and the SPRT detects this change immediately (see $t = 4.5s$ in the figure). A and B correspond to the SPRT decision making thresholds, computed based on user-defined false and missed alarm percentages.

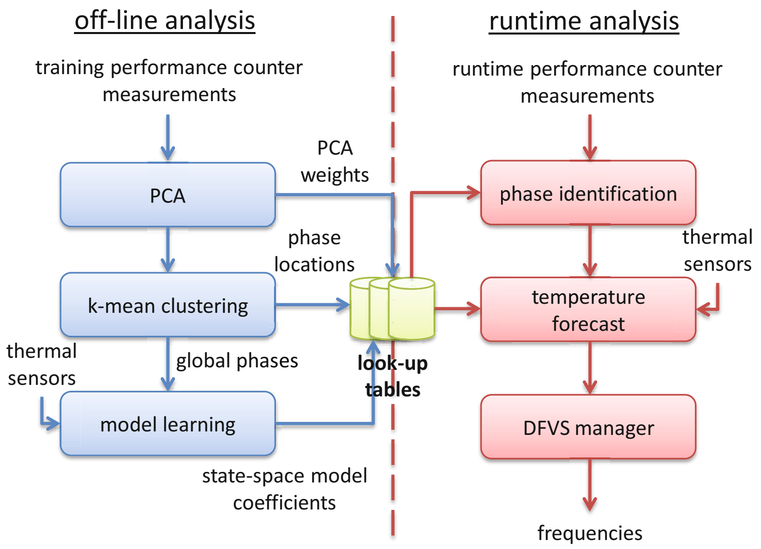


Fig. 6.7. Flow of the approach combining workload phase identification and thermal forecasting [41]

Recent work utilizes workload phase recognition in addition to temperature sensor information for low-cost, high-accuracy prediction [41]. Through principle component analysis (PCA), it is possible to determine the most relevant metrics to identify different workload phases. Then, through k-means clustering and model learning, the approach constructs a predictor offline. At runtime, the overhead is limited to identifying the phase through performance counter measurements and forecasting temperature. This forecast can then be used in thermal management decisions through tuning voltage-frequency settings. The flow of the approach is shown in Figure 6.7. Next, this chapter discusses runtime temperature measurement techniques that utilize various levels of temperature estimation and forecasting.

6.4 Runtime Thermal Management

State-of-the-art dynamic thermal management (DTM) techniques adapts to the changing workload and application environment in order to achieve the most robust performance. In general, adaptive thermal management techniques can be divided into model-based and model-free approaches. Model based thermal management learns the temperature model of the computing system. A survey on thermal modeling techniques are provided in Section 6.3. If the predicted temperature exceeds a threshold then actions will be taken to prevent hot spot. Instead of learning the temperature model, model-free adaptive thermal management adapts the control actions directly based on the collected feedback from the system. Both techniques monitor the system and extract information from recent system dynamics to develop the model either for temperature prediction or for decision making. Next, we discuss recent work in both model-based and model-free adaptive thermal management.

6.4.1 Model-Based Adaptive Thermal Management

The model-based thermal management can also be referred as predictive thermal management. These techniques make control decisions based on projected system dynamics. As long as the prediction is accurate, thermal emergencies can be avoided by taking appropriate actions in advance. Several predictive DTM policies have been proposed recently [42–45]. The main difference among these methods lies in the adopted temperature prediction models and the DTM actions. A detailed discussion of temperature prediction is provided in section 6.3. In this section we focus on the actuation based on the predictive models.

Two commonly used knobs by the DTM controllers are temperature-aware workload migration (or workload allocation) and dynamic voltage and frequency scaling (DVFS). The main goal of the DTM controller is to achieve an evenly distributed workload either spatially or temporally to limit temperature increase. We focus on several recently proposed predictive workload migration/allocation and DVFS-based techniques in this chapter.

6.4.1.1 Predictive Temperature-Aware Task Migration

In a multi-core single-chip processor, the temperature of a core is not only determined by the power consumption of the software program running on it but also on its heat dissipation ability and the temperature of its neighbors. The basic concept of temperature-aware task migration is to dynamically move "hot" processes to cores with superior heat dissipation and/or cores surrounded by cooler neighbors while moving "cool" processes to cores with inferior heat dissipation and/or cores surrounded by hotter neighbors. Most predictive thermally-aware task migration methods assume that the power consumption of a task within a given workload can be obtained either by profiling or estimation. All of them require temperature prediction models which estimate either the steady state temperature [42,44] or the temperature in future time instances [42,43].

Based on predicted temperature, Yeo et al. [42] propose a thermally-aware task migration policy called Predictive Dynamic Thermal Management (PDTM). When a task running on a processor is projected to exceed the temperature threshold, it will be moved to a processor that is predicted to be the coolest in the future. The policy is evaluated on an Intel Quad-core processor. Processor temperature can be read from the digital thermal sensors embedded in the cores. The applications running on the system are libquantum, perlbench, bzip2, hmmer from SPEC 2006 Benchmarks. The prediction model achieves very high accuracy and the temperature prediction error is only 1.6%. Compared to existing thermal management schemes, the proposed task migration policy reduces the average temperature by 7% and peak temperature by 3 °C.

Coskun et al. [43] propose a Proactive Thermal Balancing (PTB) policy. Similar to the PDTM policy [42], this policy moves the tasks from a processor core predicted to be hotter to a core that is expected to be cooler. The difference is that PDTM moves the current running tasks while PTB gives priority to moving the tasks in the waiting queue. This enables the hot core to have a period with fewer number of threads and cool down after current task finishes, and hence avoids hot spots. As mechanism of migrating a waiting task in the ready queues has already been implemented in the OS scheduler for load balancing purposes, this technique does not introduce additional implementation overhead. Experimental results on an UltraSPARC T1 based processor model show that PTB reduces hot spot occurrences, spatial gradients, and thermal cycles by 60%, 80% and 75% respectively on average compared to reactive thermal management, where workload migration decisions are made after cores reach a given threshold. In addition, PTB only incurs a performance cost of less than 2% with respect to the default scheduling policy for load balancing, as measured on real-life processors.

Both PTB and PDTM are centralized approaches. This means that these techniques require a controller that monitors the temperature and workload distribution of the entire chip and makes global decisions of resource allocation. Such centralized approaches may not scale well to a large number of cores. As the number of processing elements grows, the complexity of solving the resource management problem grows super-linearly. Furthermore, a centralized monitoring and commanding framework incurs a large overhead, as communication between central controller and cores will increase exponentially [46].

Ge et al. [44] propose a framework for distributed thermal management where a balanced thermal profile can be achieved by thermal throttling as well as thermally-aware task migrations among neighboring cores. The framework has a low-cost agent residing in each core. The agent observes the workload and temperature of local processor while communicating and exchanging tasks with its nearest neighbors. The goal of task migration is to distribute tasks to processors based on their heat dissipation capabilities and also ensure that each processor has a balanced mix of high power and low power tasks. The authors refer to the proposed technique as distributed thermal balancing migration (DTB-M) as it aims at balancing the workload and temperature of the processors simultaneously.

The DTB-M policy basically can be divided into 3 phases: temperature checking and prediction, information exchange, and task migration. Figure 6.8 shows the flowchart of the DTB-M execution in the *i*th core. A DTB-M agent is initially neutral. It will enter the master mode if any of the three scenarios are true: (1)The local temperature reaches a threshold T_m (in this case, the DTB will first stall the processor to let it cool down before it enters the master mode), (2) the predicted future peak temperature exceeds the threshold, (3) the temperature difference of the local core and the neighbor core exceeds the thermal balancing threshold. Otherwise, it will enter the slave mode. A master DTB agent issues a task migration request to its nearest neighbors which are DTB slaves.

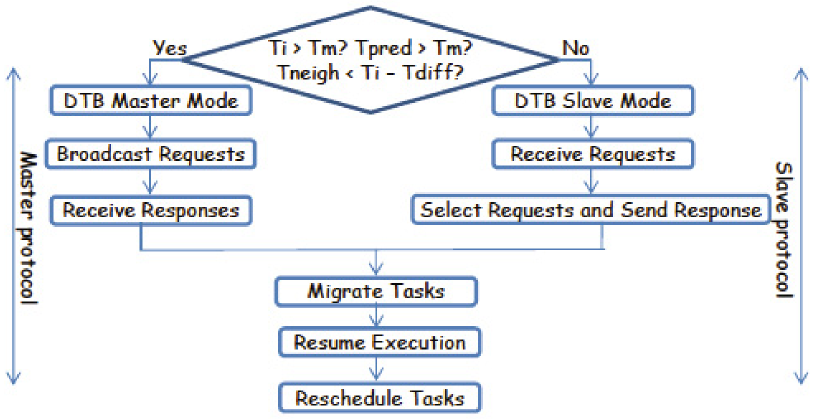


Fig. 6.8. Flow of the Distributed Thermal Balancing-Migration (DTB-M) technique

The DTB-M policy has two components. Both of them have quadratic complexity with respect to the number of tasks in the local task queue. The first component is a steady state temperature based migration policy (SSTM). It considers the long term thermal behavior of tasks, and distributes tasks to cores based on their different heat dissipation ability. The second component of DTB-M is a temperature prediction based migration policy (TPM), which predicts the peak temperatures of different task combinations when making migration decisions. It ensures that each core executes a balanced mixture of high power and low power tasks without having thermal emergencies.

These two components are complementary to each other as the SSTM considers long term average thermal effect and the TPM considers short term temporal variations. Experimental results show that the DTB-M has 66.79% lower frequency of hot spots and 40.21% higher performance compared to the existing techniques. Furthermore, DTB-M also has much lower migration overhead. The number of migrations is reduced by 33.84% while the overall migration distance is reduced by 70.7% due to the fact that processors only communicate and exchange tasks with their nearest neighbors.

6.4.1.2 Predictive DVFS for Thermal Management

Cochran et al. [41] utilize the program internal characteristics to predict future temperature. We know today that the execution of a program typically includes multiple distinct phases. A program phase is a stage of execution in which a workload exhibits near identical power, temperature or performance characteristics. Therefore, as long as we are able to identify the program phases during the run time and construct the relation between operating temperatures and program phases, we can predict the future temperature of an application.

The run time characteristics of a program can be characterized by architectural events such as instructions per cycle (IPC), memory access rate, cache miss rate and floating point (FP) instructions executed, etc. After selecting the most relevant events, the next step is to group similar program phases together as similar behavior will produce similar operating temperature. This process is carried out on a training data set with data points which are extracted from a set of representative workloads. The aim of the grouping process is to classify the n data points to k clusters such that the sum of the square distances between each training data point to its cluster center is minimized. Then a linear model is used to represent the relation between temperature and the workload phase. The model is trained using least square fitting. At every regular time interval, the event counters information is collected and the temperature model is evaluated for every available frequency setting. The highest frequency which will not cause thermal violation is selected to run for the next interval. Experimental results show that, compared to a reactive DVFS based thermal management policy, this algorithm cuts down the thermal violation considerably while reducing the runtime by 7.6%.

While all of the above mentioned techniques aim at reducing the temperature hot spots as the main objective, Bartolini et al. [45] propose to perform DVFS for both energy minimization and temperature management. A decoupled energy and thermal controller is introduced in this work. Based on the estimated clock cycles per instruction (CPI) count and performance constraints, the energy controller calculates the optimal frequency trajectory ($fEC(t)$) that minimizes energy while meeting the performance requirements. The thermal controller selects the clock frequency that has the least deviation from $fEC(t)$ while keeping the core temperature below the given threshold. The thermal controller predicts the future temperature using an ARX (AutoRegressive eXogenous) model, which is derived from a self-calibration routine.

6.4.1.3 Model-Free Adaptive Thermal Management

While model-based DTM selects action based on predicted temperature, model-free thermal management learns the actions directly. Ge et al. [47] propose to apply reinforcement learning (RL) to the DTM problem for multimedia applications. They model the DTM problem as a stochastic control process and adopt the RL algorithm to find the optimum policy during runtime. They model the processor's DTM controller as a learning agent and consider the rest of the system as the environment. After the learning agent takes an action, it observes the environment and estimates the reward or penalty induced by this action. The agent learns from this experience and tries to improve its future decisions to maximize the reward through the learning model. The proposed learning model has very small run time overhead. It only incurs a few table lookups and some simple arithmetic operations. Compared to a policy without thermal management, this learning policy reduces thermal violations by 34.27% while maintaining similar run times.

Coskun et al. [48] propose a switching experts based learning algorithm for thermal management. Their learning technique leverages a set of management policies, including dynamic power management to turn off idle cores (DPM), DVFS, thread migration and load balancing. These policies are referred to as experts. On top of the experts, there are specialists which are higher level policies that determines which expert to use in the next interval. Each specialist is assigned with a weight which is a function of performance and temperature. At each time interval, only specialists that select the active expert have their weight updated. The specialist that has the highest weight is selected and the corresponding expert policy is executed. Through updating weights for the specialists, the technique guarantees to converge to the policy that is best-fitting to current workload dynamics. Experimental results show that this learning technique is able to improve system thermal behavior, reduce energy, and increase performance significantly compared to the default scheduling policy in the OS.

6.4.1.4 Learning Based Thermal Management at Machine and Server Level

Learning-based DTM techniques are not restricted to microprocessors and could be applied at a higher level, such as a data center. For a data center, the major problem is to keep the temperature of the server chassis under a threshold while reducing the computing power and the power consumption of the cooling system, e.g., cooling fan or Computer Room Air Condition (CRAC), as much as possible. Pakbaznia et al. [49] propose a workload placement algorithm for large data centers. Their algorithm relies on an exponential workload requests predictor to forecast the future income requests to a data center. The rationale behind this prediction model is that the data center workloads typically show a repetitive pattern with a period in the order of hours, days, weeks and so forth. The accuracy of the prediction model is high: the predicted incoming requests are within 5% of the real incoming requests. The technique employs Integer Linear Programming (ILP) to allocate the workload and turn on or off specific servers to minimize the total data center energy. Experimental results show that their algorithm achieves consistent energy savings compared to greedy algorithms during a 24 hour simulation.

6.5 Conclusions

Thermal Management is an interesting and challenging design problem for the next generation processors. Thermal gradients are increasing with each new technology generation, forcing designers to think outside the traditional realm. Each of the design phases that were discussed in this chapter are interdependent and support the dynamic thermal management. In the future generation processors, the prediction and response mechanisms can further benefit from using evolutionary based algorithms for thermal management. For example, the run-time controller to manage the different thermal policies uses complex non-linear models and the time constants involved in arriving at accurate solutions is large. Designers can look at lean algorithms that provide faster and simpler numerical solutions. Such algorithms can benefit by abstracting design coefficients from different hierarchies, including thermal, electrical and system level. Another critical trend in the future generation processors is the use of 3D integration, with JEDEC [50] standards established recently. Thermal management in these systems requires an awareness of the physical behavior of the different stacks, associated cooling mechanisms, temperature control mechanisms that interface across the stacks, and algorithms that optimally distribute or allocate resources based on the thermal gradients.

Acknowledgements. The authors would like to thank Katherine E. Dellaquila from Hewlett Packard for her contributions on the thermal sensor placement mechanisms section.

References

1. Alley, R., Soto, M., Kwark, L., Crocco, P., Koester, D.: Modeling and validation of on-die cooling of dual-core cpu using embedded thermoelectric devices. In: Twenty-fourth Annual IEEE Semiconductor Thermal Measurement and Management Symposium, Semi-Therm 2008, pp. 77–82 (March 2008)
2. Memik, S.O., Mukherjee, R., Ni, M., Long, J.: Optimizing Thermal Sensor Allocation for Microprocessors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(3), 516–527 (2008), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4454017
3. Viswanath, R., Wakharkar, V., Watwe, A., Lebonheur, V.: Thermal performance challenges from silicon to systems. Intel Technology Journal Q3, 1–16 (2000)
4. Xiang, Y., Chantem, T., Dick, R.P., Hu, X.S., Shang, L.: System-level reliability modeling for mpsocs. In: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES/ISSS 2010, pp. 297–306 (2010), http://doi.acm.org/10.1145/1878961.1879013
5. Borkar, S.: Design challenges of technology scaling. IEEE Micro 19(4), 23–29 (1999)
6. Gronowski, P., Bowhill, W., Preston, R., Gowan, M., Allmon, R.: High-performance microprocessor design. IEEE Journal of Solid-State Circuits 33(5), 676–686 (1998)
7. Skadron, K., Huang, W.: Analytical model for sensor placement on microprocessors. In: 2005 International Conference on Computer Design, pp. 24–27 (2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1524125
8. Mukherjee, R., Memik, S.O.: Systematic temperature sensor allocation and placement for microprocessors. In: Proceedings of the 43rd Annual Design Automation Conference, DAC 2006, pp. 542–547. ACM, New York (2006)
9. Kwasinski, A., Kudithipudi, D.: Towards integrated circuit thermal profiling for reduced power consumption: Evaluation of distributed sensing techniques. In: Proceedings of the International Conference on Green Computing, GREENCOMP 2010, pp. 503–508. IEEE Computer Society, Washington, DC (2010)
10. Yun, X.: On-Chip Thermal Sensor Placement, Master's, University of Massachusetts Amherst (2008), http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1242&amp;context=theses
11. Dellaquila, K.: Thermal Profiling of Homogeneous Multi-Core Processors Using Sensor Mini-Networks, Master's, Rochester institute of Technology (2010), https://ritdml.rit.edu/bitstream/.../KDellaquilaThesis8-2010.pdf?...1
12. SPEC-CPU 2000, Standard Performance Evaluation Council, Performance Evaluation in the New Millennium, Version 1.1 (2000)
13. Skadron, K., Lee, K.: Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors. In: 19th IEEE International Parallel and Distributed Processing Symposium, pp. 232a–232a (2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1420152
14. Burger, D., Austin, T.: SimpleScalar Tutorial. In: 30th International Symposium on (1997), http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:SimpleScalar+Tutorial#2
15. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
16. Long, J., Memik, S., Memik, G., Mukherjee, R.: Thermal monitoring mechanisms for chip multiprocessors. ACM Transactions on Architecture and Code Optimization 5(2), 1–33 (2008), http://portal.acm.org/citation.cfm?doid=1400112.1400114
17. Skadron, K., Stan, M., Huang, W., Velusamy, S.: Temperature-aware microarchitecture: Extended discussion and results. University of Virginia, Department of Computer Science (2003), http://scholar.google.com/scholar?q=intitle:Temperature-Aware+Microarchitecture:+Extended+Discussion+and+Results#0

18. Sabuncu, M.R., Ramadge, P.J.: Gradient based nonuniform subsampling for information-theoretic alignment methods. In: 26th Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (IEMBS), pp. 1683–1686 (2004)
19. Cochran, R., Nowroz, A.N., Reda, S.: Post-silicon power characterization using thermal infrared emissions. In: ISLPED, pp. 331–336 (2010)
20. Mesa-Martinez, F.J., Nayfach-Battilana, J., Renau, J.: Power model validation through thermal measurements. In: ISCA, pp. 302–311 (2007)
21. Pedram, M., Nazarian, S.: Thermal modeling, analysis, and management in vlsi circuits: Principles and methods. Proceedings of the IEEE 94(8), 1487–1501 (2006)
22. Isci, C., Martonosi, M.: Runtime power monitoring in high-end processors: Methodology and empirical data. In: Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 36, p. 93 (2003)
23. Khan, M.A., Hankendi, C., Coskun, A.K., Herbordt, M.C.: Software optimization for performance, energy, and thermal distribution: Initial case studies. In: IEEE Workshop on Thermal Modeling and Management: From Chips to Data Centers (in conj. with Green Computing Conference (IGCC)) (2011)
24. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: a framework for architectural-level power analysis and optimizations. In: ISCA, pp. 83–94 (2000)
25. Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M., Jouppi, N.P.: Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In: MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 469–480 (2009)
26. Skadron, K., Stan, M., Huang, W., Velusamy, S., Sankaranarayanan, K., Tarjan, D.: Temperature-Aware Microarchitecture. In: ISCA, pp. 2–13 (2003)
27. Skadron, K., Stan, M.R., Sankaranarayanan, K., Huang, W., Velusamy, S., Tarjan, D.: Temperature-aware microarchitecture: Modeling and implementation. In: TACO, vol. 1(1), pp. 94–125 (2004)
28. Atienza, D., Valle, P.D., Paci, G., Poletti, F., Benini, L., Micheli, G.D., Mendias, J.M.: A Fast HW/SW FPGA-Based Thermal Emulation Framework for Multi-Processor System-on-Chip. In: Design Automation Conference (DAC), pp. 618–623 (2006)
29. Coskun, A.K., Atienza, D., Rosing, T.S., Brunschwiler, T., Michel, B.: Energy-efficient variable-flow liquid cooling in 3d stacked architectures. In: DATE, pp. 111–116 (2010)
30. Link, G.M., Vijaykrishnan, N.: Thermal trends in emerging technologies. In: Proceedings of the 7th International Symposium on Quality Electronic Design, ISQED 2006, pp. 625–632. IEEE Computer Society, Washington, DC (2006)
31. Sridhar, A., Vincenzi, A., Ruggiero, M., Atienza, D., Brunschwiler, T.: 3d-ice: Fast compact transient thermal modeling for 3D-ICs with inter-tier liquid cooling. In: International Conference on Computer-Aided Design, ICCAD 2010 (2010)
32. Huang, W., Stan, M.R., Skadron, K., Sankaranarayanan, K., Ghosh, S., Velusam, S.: Compact thermal modeling for temperature-aware design. In: Proceedings of the 41st Annual Design Automation Conference, DAC 2004, pp. 878–883. ACM, New York (2004)
33. Velusamy, S., Huang, W., Lach, J., Stan, M.R., Skadron, K.: Monitoring temperature in fpga based socs. In: ICCD, pp. 634–640 (2005)
34. Sridhar, A., Vincenzi, A., Ruggiero, M., Brunschwiler, T., Atienza Alonso, D.: Compact transient thermal model for 3D ICs with liquid cooling via enhanced heat transfer cavity geometries. In: Proceedings of the 16th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC 2010), vol. 1(1), pp. 105–110. IEEE Press, New York (2010)
35. Hung, W.-L., Link, G.M., Xie, Y., Vijaykrishnan, N., Irwin, M.J.: Interconnect and thermal-aware floorplanning for 3d microprocessors. In: ISQED, pp. 98–104 (2006)
36. Sharifi, S., Rosing, T.V.: Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management. Trans. Comp.-Aided Des. Integ. Cir. Sys. 29, 1586–1599 (2010)

37. Cochran, R., Reda, S.: Spectral techniques for high-resolution thermal characterization with limited sensor data. In: DAC, pp. 478–483 (2009)
38. Coskun, A.K., Rosing, T.V., Gross, K.C.: Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs. IEEE Transactions on CAD 28, 1503–1516 (2009)
39. Yeo, I., Liu, C.C., Kim, E.J.: Predictive dynamic thermal management for multicore systems. In: DAC, pp. 734–739 (June 2008)
40. Coskun, A.K., Rosing, T., Gross, K.: Proactive Temperature Balancing for Low-Cost Thermal Management in MPSoCs. In: International Conference on Computer-Aided Design (ICCAD), pp. 250–257 (2008)
41. Cochran, R., Reda, S.: Consistent runtime thermal prediction and control through workload phase detection. In: Design Automation Conference, DAC (2010)
42. Yeo, I., Liu, C.C., Kim, E.J.: Predictive dynamic thermal management for multicore systems. In: Proceedings of the 45th Annual Design Automation Conference, DAC 2008, pp. 734–739. ACM, New York (2008), http://doi.acm.org/10.1145/1391469.1391658
43. Coskun, A., Rosing, T., Gross, K.: Utilizing predictors for efficient thermal management in multiprocessor socs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28(10), 1503–1516 (2009)
44. Ge, Y., Malani, P., Qiu, Q.: Distributed task migration for thermal management in many-core systems. In: 2010 47th ACM/IEEE on Design Automation Conference (DAC), pp. 579–584 (June 2010)
45. Bartolini, A., Cacciari, M., Tilli, A., Benini, L.: A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1–6 (March 2011)
46. Ebi, T., Faruque, M., Henkel, J.: Tape: Thermal-aware agent-based power econom multi/many-core architectures. In: IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers, ICCAD 2009, pp. 302–309 (November 2009)
47. Ge, Y., Qiu, Q.: Dynamic thermal management for multimedia applications using machine learning. In: 2011 48th ACM/EDAC/IEEE on Design Automation Conference (DAC), pp. 95–100 (June 2011)
48. Coskun, A., Rosing, T., Gross, K.: Temperature management in multiprocessor socs using online learning. In: 45th ACM/IEEE on Design Automation Conference, DAC 2008, pp. 890–893 (June 2008)
49. Pakbaznia, E., Ghasemazar, M., Pedram, M.: Temperature-aware dynamic resource provisioning in a power-optimized datacenter. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 124–129 (March 2010)
50. JEDEC, 3D IC Standards (2011), http://www.jedec.org/standards-documents/technology-focus-areas/3d-ics