Estimation of thermal status in multi-core systems

Simone Corbetta and William Fornaciari
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Via Ponzio 34/5, 20133 Milano ITALY
Email: {scorbetta,fornacia}@elet.polimi.it

***Abstract*— Modern multi-core architectures are prone to a complex dynamic thermal management process: the presence of multiple cores adds challenges to the temperature estimation activity, due to the induced heat-up process from adjacent cores. Run-time estimation can benefit from floorplan information to better estimate the thermal characteristics of each core, and transient information can help the system to predict and avoid thermal alarms, increasing the system reliability and lifetime. In this context, the present work aims at defining a novel on-line measurement methodology based on neighbor nodes and transient information, providing a metric to be employed in thermal-aware designs, either at design-time to characterize application and platform from a thermal view-point, or at run-time in conjunction with the Dynamic Thermal Management subsystem. The proposed methodology intercepts floorplan-induced thermal behavior that would be otherwise unrecognized, and it also shows how a non-floorplan-aware methodology can reveal up to a 30% error in the estimate of thermal status.**

I. Introduction

Thermal-induced reliability issues are of major importance in modern electronic systems. It has been demonstrated that more than 50% of all IC failures are due to thermal issues [1]. Temporal variations of temperature are generally unwanted, since high-range thermal variations affect device reliability, lifetime duration and stress the package, possibly leading to break-down. In addition, gradients of temperature arise across the chip due to the different workload in different portions of the same chip. This last aspect is particularly true in high-complexity MPSoCs, where independent subsystems exist and multiple applications run at the same time.

The temperature profile of an electronic system can present a wide range of variability along its lifetime, and this variability basically depends on the workload and input data. In a single-core environment core temperature is basically tied to the sole core power/performance configuration (operating frequency and supply voltage) and its activity (whether or not an application is running). Temperature is just an effect of the self-heating process, due to power consumption. However, this is no longer true in multi-core architectures, where thermal coupling effects arise, and such effects increase with the number of integrated cores per chip. The two columns in Fig. 1 demonstrate how the temperature profile of the cores in an Intel i7 quad-core processor depends on the workload and on the thermal-coupling effects from adjacent cores. In particular, it can be seen how even a non computation-intensive workload (as it happens in Core 2 in scenario (a) or in Core 3 in scenario (b)) is associated with a relatively high temperature (indeed, the temperature of the two cores is quite high). This aspect should be taken into consideration explicitly while modeling the effects of temperature on reliability, since otherwise an incorrect thermal status would be recognized, mining the reliability concerns of the entire system. In this perspective, another aspect that is of utmost relevance comes from transients. The rate at which a core heats up depends on the difference between the steady-state temperature and the current core temperature, the physical parameters constraining the thermal coupling effects, the workload of the core and of the neighbor ones, and the power consumption level. The awareness of the current position of the core on the thermal transient profile gives us the advantage of preventing potential thermal alarms, increasing the overall system reliability and lifetime.

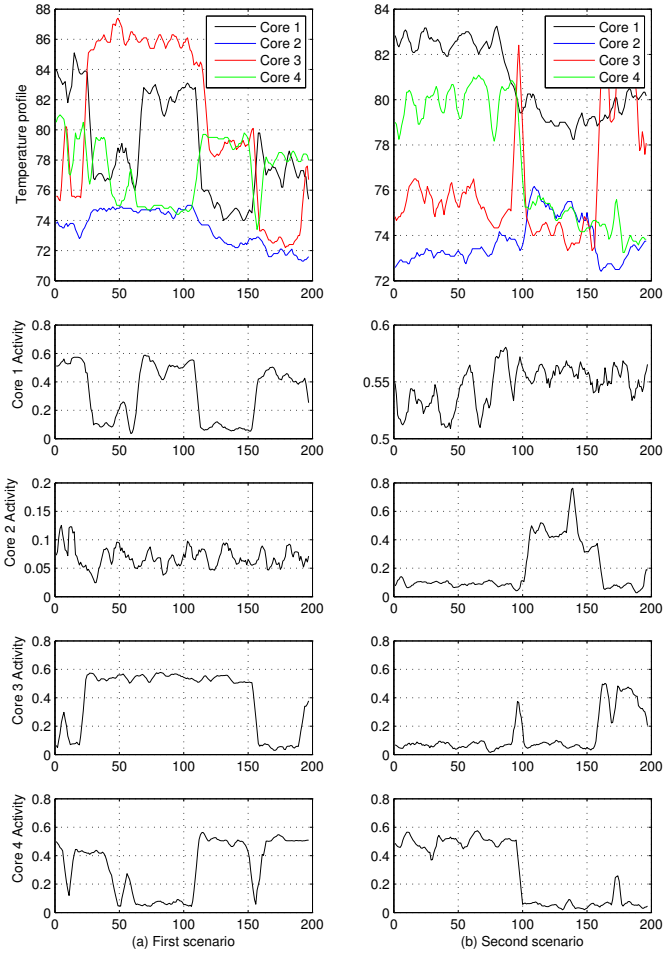

Fig. 1. Thermal coupling effects in a quad-core architecture.

A. Related works and novel contributions

Several approaches have been presented in literature for the management of thermal profiles within a single-chip multiprocessor (CMP). One of the main aspects of the Dynamic Thermal Management (DTM) approaches relies on the estimation procedure used to update the thermal status of the core, information used by the manager to actuate power and thermal related decisions.

At the best of our knowledge, the first work on predictive thermal management has been proposed in [2]. A simple predictive control algorithm computes the maximum frequency allowed for the system, considering the thermal envelope as the main metric. The objective is to reconfigure the system avoiding thermal alarms. The control

978-1-4244-9474-3/11/$26.00 ©2011 IEEE

algorithm configuration selection policy is based on an off-line profiling of the applications. However, this approach targets only local thermal profile and does not take into account thermal coupling effects between adjacent cores.

The approach in [3] predicts the future temperature on a history-based process. At run-time, the predicted temperature is a weighted function of the current temperature and the current workload. Neither such an approach considers the heat transferred to/from other adjacent cores due to the asymmetric workload distribution across the chip, and the system configuration $\langle V, f \rangle$.

Multi-core systems (MPSoCs mainly) are taken into consideration in [4], in which the prediction is based on an ARMA model. The model dynamically adjusts the coefficients value to follow the changing execution context at run-time, based on the prediction error. However, no real temperature trace has been taken into consideration, but simulation results. This last aspect should be considered since real traces are the true result of physical coupling (from the thermal view-point) of the different subsystem, and these results cannot be known in advance. In addition, no floorplan information is taken into account negatively impacting on the usability of the model itself.

Last, [5] proposes a simplified ARMA model based on the observation that temperature spectrum is a band-limited signal, so that the Moving Average contribution can be easily neglected from the model. The approach considers predictor coefficients that are set once for all at design-time, according to the band-limited spectrum profile. Thus, there is no dynamic adaptation of the coefficients, with no focus on changing conditions and spatial temperature influence.

According to what has been said in Section I and according to the state-of-the-art in DTM, our research work is based on the observation of how thermal-related issues are highly influenced by the floorplan of the device (i.e., the physical distance between adjacent cores), and how transients give us lots of information about the thermal profile. In this perspective, the novel contribution of the proposed work stands in a methodology for on-line estimation of the thermal status taking into consideration the effects of the floorplan on the system thermal profile, while preventing thermal alarms through a transient-based modeling. For this purpose, a new metric has been defined representing the *thermal status* of each core, and a rate index at which it is estimated to heat-up.

The rest of the paper is organized as follows: the formal model for the estimation of the thermal status is given in Section II, and the Matlab prototype is then discussed in Section II-A. Experimental results are shown in Section III, and conclusions are drawn in Section IV.

II. Proposed methodology

The heat associated with a core has three contributions. A *generative* contribution accounts for the workload. This is the self-heating contribution, and is independent on the floorplan. A *coupling* contribution instead takes into account thermal coupling effects of neighbor cores, and it is basically a function of the distance between cores. This term models the spatial aspect of thermal heat diffusion. Last, a final *convective* contribution comes from the heat exchange process with the environment, either in a forced manner through fans or through packaging with cooler ambient air.

The formal model is reported in Equation 1; all the aforementioned parameters are collected and integrated in a single expression $s_j(t)$ (the *thermal status*) for a generic core j at time t.

$$s_j(t) = \alpha_j(t) g_j(t) + \sum_{k \in N_j} \frac{1}{d_{jk}} \alpha_k(t) g_k(t) - (c_1 + c_2) \qquad (1)$$

The first term $g_j(t)$ is the generative contribution. Its definition is given in Equation 2. The core's power/performance configuration is specified through its voltage and frequency values; to simplify the on-line estimation process, the power/performance configuration is defined by means of the maximum voltage V_{MAX} and frequency f_{MAX} values. $w_j(t)$ represents an index of the activity of that core, and $T_j(t)$ its absolute temperature. The additive term $exp\{-1/T_j(t)\}$ takes into account temperature-dependent leakage effects. The values ξ_{dyn} and ξ_{st} are used to weight the dynamic and static power contributions; their typical values are set according to the specifications in [6] to reflect technology, e.g. $\langle \xi_{dyn}, \xi_{st} \rangle = \langle 0.65, 0.35 \rangle$. In the thermal status expression in Equation 1, the generative value is further weighted through an index $\alpha_j(t)$ accounting for aging/reliability aspects. The value of α changes over time [7], as a consequence of the temperature-induced reliability concerns, and can be considered constant for simplicity ($\alpha_j(t) = 1$ for each t).

$$g_j(t) \propto \xi_{dyn} \cdot \left(\left(\frac{V_j(t)}{V_{MAX}} \right)^2 \frac{f_j(t)}{f_{MAX}} w_j(t) \right) + \\ + \xi_{st} \cdot exp\{-1/T_j(t)\} \qquad (2)$$

The second contribution in Equation 1 is due to the neighbors activity. The finite summation is extended to each core k that falls within the neighborhood set N_j of core j, i.e. all those cores that are thermally coupled with it. d_{jk} is the distance between the generic core j and k, while $g_k(t)$ has the same meaning as above. The last contribution is due to the heat transfer: c_1 for forced convective contribution due to fans, while c_2 for the heat released to ambient through the package. Without loss of generality, we can consider these factors as constants in our modeling ($c_1 = c_2 = 0$).

The status $s_j(t)$ models the thermal status of the core during the temperature transient. As it has been aforementioned, the point where the core is currently located on the transient is crucial to reveal the future behavior of the core itself, based on the distance from the steady-state temperature, and on its current status: the transient follows a sigmoid-like profile (see at the temperature profile of one of the cores in Fig. 1), so that the derivative decreases asymptotically to 0 when the target temperature is reached. With this in mind, a valuable metric defining the heat-up rate is reported in Equation 3. $\Omega_j(t)$ represents the index of the heat-up rate for the generic core j at time t; the term between the parentheses is the signed difference between the thermal status that core j would fall in if the temperature is set to the predefined threshold $threshold\_status_j(t)$, and its current generative contribution $g_j(t)$. In the former term, we explicitly used the sole generative contribution instead of the one with neighbor influence since we are interested in an estimate of the rate, and even though the presence of neighbor impacts on the rate (by increasing its mean value), the rate is major relying on the generative term. $\rho = (T_{steady} - T_{curr})$ is the *steady-state distance coefficient*, and weights the rate according to the current distance from the steady-state temperature. Such rate gives us a way to focus on the temperature profile trend, and to obtain additional useful information from the execution.

$$\Omega_j(t) = \rho \cdot (threshold\_status_j(t) - g_j(t)) \qquad (3)$$

A few words have to be said regarding the value of Ω. Since it is an index, it can be conveniently represented as a percentage value, so the output has a relative meaning. It can be employed by the thermal manager to take a thermal-optimizing decision, but it should have a reference value. To this purpose, it can compare

the metric value with a reference one provided a threshold value T_{thr}; the metric can compare the values of the rate in the two cases and take decisions accordingly. Thus, provided that $\widehat{\Omega}_j(t) = \rho \cdot (steady\_status_j(t) - g_j(t))$ is for the steady-state case and $\Omega_j(t) = \rho \cdot (threshold\_status_j(t) - g_j(t))$ is for the threshold case, the effective value $\theta_j(t)$ for the metric can be defined as $\theta_j(t) = \frac{\Omega_j(t)}{\widehat{\Omega}_j(t)}$.

A. The estimation process

In the previous section, we provided the ingredients for our estimation process, while in this section we aim at providing an overview of the algorithmic representation. For our experimental purposes, data has been collected in advance and then a Matlab prototype algorithm has been used to generate output data and the reference thermal status.

The operating point can be determined by reading the current supply voltage and the frequency settings. Both values can be easily determined at the operating system level (through ACPI or ad-hoc interfaces); however, since not all the architectures have accessible supply voltage levels registers, we used the datasheet of the target architecture to get the typical value for the corresponding frequency [8]. Temperature sensors are also generally employed in modern systems. Last, a valid approach to monitor CPU activity is to consider the readings from hardware performance counters, and calibrating such readings to the effective running workload [9]. This can be done by reading from OS-specific memory regions.

The algorithm is reported in Algorithm 1. Input parameters are the maximum operating frequency f_{MAX} and supply voltage V_{MAX}, the set of available Operating Points $OP_j = \langle V_j, f_j \rangle$, the matrix of distances between cores D and the weights ξ_{dyn} and ξ_{st} for the dynamic and static power consumption contribution. As output, the estimate Ω of the rate. The reported pseudo-code is meant to be run from each core during the entire execution time; `SAMPLING_PERIOD` determines the sampling periods. This value can be either statically defined or dynamically adjusted according to the actual temperature profile and workload characteristics. In our experiments, we have set it to 1 second to achieve a very fine grain analysis. The very first part of the algorithm from line 2 to 8 is data collection and generative contribution term computation. The thermal status is then initialized in line 9 to the sole generative contribution. Its value is then updated in the `for` loop of lines 10-13 according to the distance between adjacent cores. In the last part, the algorithm computes the rate estimate (lines 14 and 15) according to Equation 3. This value is then returned. The major challenge in the proposed algorithm scheme is collecting data from the neighbors and send data to them. As a matter of fact, the assumption in line 12 is that g is exchanged among adjacent cores. The solution to this problem is still under development, as explained in Section IV.

Algorithm 1: Heating rate estimation

```
1  wait(SAMPLING_PERIOD);
   // Collect data from sensors
2  curr_temp ← read_sensor();
3  curr_freq ← read_frequency();
4  curr_volt ← get_voltage(freq);
   // Power/performance factor
5  ppf ← (curr_volt/V_MAX)^2 · curr_freq/f_MAX;
6  leakage_index ← exp(−1/curr_temp);
7  workload ← get_average_workload();
8  g ← ξ_dyn · ppf · workload + ξ_st · leakage_index;
   // Contribution from neighbors
9  status ← g;
10 for each neighbor j do
       // Distance from core j
11     d ← distance_matrix[j, this_core];
12     status ← status + 1/d · g(j);
   // Compute Ω
13 Δ ← T_steady − curr_temp;
14 Ω ← Δ · (threshold_status − g);
15 return Ω;
```

III. Results

The purpose of this section is to highlight the behavior of the proposed model in a real use-case. The results show how the proposed approach effectively intercepts thermal behavior that would be otherwise unrecognized. The target test platform is a quad-core Intel i7-820QM processor running at 1.73GHz, technology process of 45nm, and a Thermal Design Power of 45W [8]. The processor has 4 physical cores, but supports 8 logical cores due to the Hyper Threading technology [8]. Data related to temperature, frequency and workload is taken during a normal job session, in which different applications are mixed (e.g., kernel compilation, audio/video decoding, text editing, etc.). One data value is taken every second of activity.

As explained in Section I-A, the existing approaches for dynamic thermal management are based on an isolated view of the problem, since no coupling is considered. The model presented in Section II takes directly into account this aspect: the output is a weighted sum of contributions from the self-heating process and the thermal coupling effects. The output profile of the algorithm is shown in Fig. 2. Due to the lack of space in this paper, the profile is reported only for Core 2 from Fig. 1, but similar results are obtained for the other cores in the system, with similar discussion.

The first (top) figure reports the temperature profile of the core through a 8000-samples interval (approximately more than 2 hours of sampling). The high variability of the temperature is due to two facts mainly: the high variability of the CPU activity (influenced by the load-balancing scheduler), and the variability of the temperature (and activity) of adjacent cores. The second plot shows the core activity profile. The reported workload is relevant in its variability and alternation of busy and free bursts. Notice that in proximity of high activity periods the temperature of the core increases from its current value; this last value is directly influenced by adjacent cores, such that it demonstrates that the average temperature of the cores is lifted up due to the sole thermal coupling. The third figure shows the thermal status profile, as computed by Equation 1. In this case, we can see how the thermal status is a conjunct expression of the generative contribution (due to core's activity) and of the current temperature (influenced by thermal-coupling effects). Last, the bottom figure shows the rate profile, according to Equation 3. What we expected is that the heat-up rate would follow temperature profile in a "mirrored" sense; indeed, the heat-up rate is maximum at the transient base and decreases toward the steady-state temperature. This is effectively captured by the proposed model, and reported in the bottom plot.

The error in neglecting the neighbor effects while determining the thermal profile can be as much as 30%. Refer to Fig. 3 for an overview of how the error goes with temperature and activity contributions. As it can be seen from the figure, the error is greater

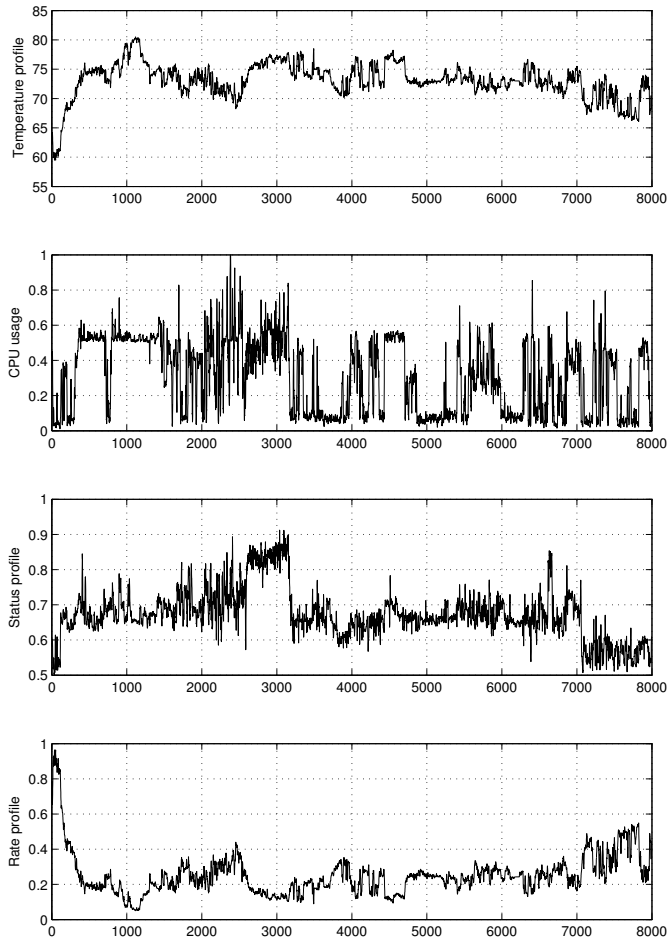

Fig. 2. Status and rate profiles for Core 2, as computed by the model presented in Section II-A.

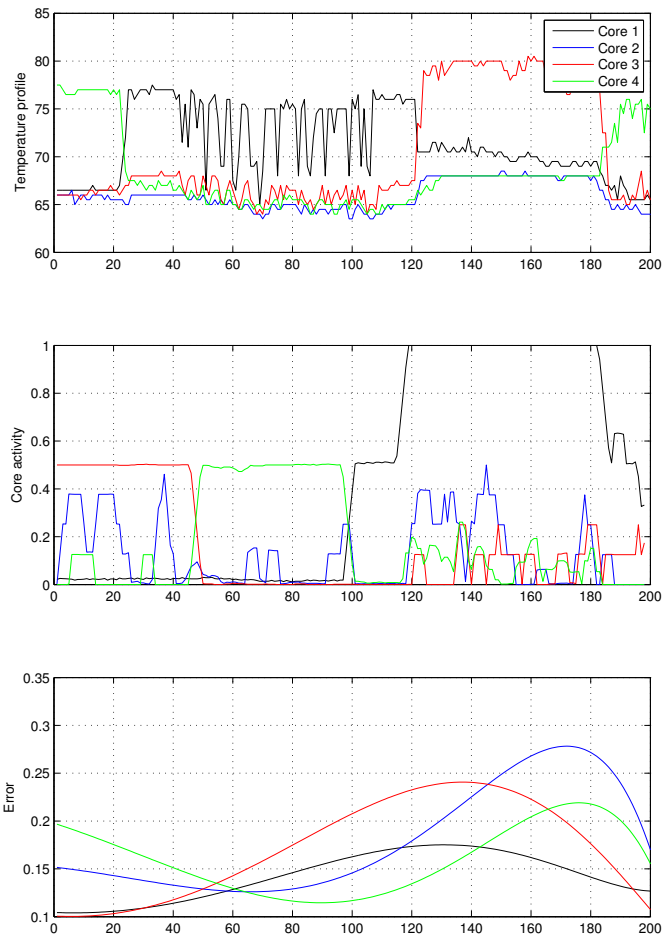

Fig. 3. Absolute error for a 200 seconds interval, for each core.

when the core has a low activity but high temperature. This is for instance what happens in the interval $[120, 180]$ for Core 2 and Core 4, whose temperatures are directly influenced by the high activity of Core 1. On the contrary, when a core has a high activity, the error diminishes since the main contribution to temperature comes from the generative one. The error in this last case accounts for the shifting effect of the average temperature while thermal-coupling effects are considered.

IV. Conclusions and Future Works

The goal of this research work, supported by 2PARMA (http://www.2parma.eu) and COMPLEX (http://complex.offis.de) FP7 European projects, has been the definition of a suitable methodology for on-line temperature estimation. The work is focused on a metric representing the thermal status of the cores in a multi-core environment, and experimental results show how floorplan information is very helpful for a better estimation of the thermal behavior at run-time, decreasing the error whose value can reach up to 30%. The proposed methodology can be employed at design-time (in EDA tools), to characterize the power/thermal behavior of the target application and platform, or at run-time by a Manager to jointly manage power, performance and thermal aspects.

Future research work is focused on the integration of the proposed model with the DTM subsystem, the design and development of an ad-hoc infrastructure to exchange thermal information among cores. A viable and interesting solution to this last aspect will be addressed through an appropriate Network-on-Chip monitoring architecture.

References

[1] M. Pedram and S. Nazarian, "Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods," *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1487–1501, August 2006.

[2] J. Srinivasan and S. V. Adve, "Predictive dynamic thermal management for multimedia applications," in *ICS '03: Proceedings of the 17th annual international conference on Supercomputing*. New York, NY, USA: ACM, 2003, pp. 109–120.

[3] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *DAC '08: Proceedings of the 45th annual Design Automation Conference*. New York, NY, USA: ACM, 2008, pp. 734–739.

[4] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Proactive temperature management in MPSoCs," *International Symposium on Low Power Electronics and Design*, 2008.

[5] R. Z. Ayoub and T. S. Rosing, "Predict and act: dynamic thermal management for multi-core processors," *International Symposium on Low Power Electronics and Design*, 2009.

[6] International Technology Roadmap for Semiconductor, "Design chapter," 2009.

[7] JEDEC, "Failure Mechanisms and Models for Semiconductor Devices," Arlington, 2009.

[8] Intel Corporation, "Intel©core™i7-900 mobile processor extreme edition series, intel core i7-800 and i7-700 mobile processor series datasheet," September 2009.

[9] R. Knauerhase, P. Brett, B. Hohlt, T. Li, and S. Hahn, "Using OS Observations to Improve Performance in Multicore Systems," in *IEEE Micro*, vol. 28, no. 3, 2008, pp. 54–66.