Exploiting Thermal Transients With Deterministic Turbo Clock Frequency

Pierre Michaud

Exploiting Thermal Transients With Deterministic Turbo Clock Frequency

Pierre Michaud
Inria, Univ Rennes, CNRS, IRISA
Campus universitaire de Beaulieu, 35000 Rennes, France
pierre.michaud@inria.fr

February 2020

Abstract

Modern microprocessors feature turbo mechanisms that adjust the clock frequency dynamically so as to maximize processor performance under power and temperature limits. However, the documentation for commercial chips rarely provides more than a superficial description of how turbo works. This paper highlights certains aspects of turbo that are not well known outside the industry and that distinguish it from dynamic thermal management. A plausible open-source turbo is proposed and described, extrapolating from the scarce and sparse information that has been disclosed so far.

1 Introduction

The clock frequency of modern multicore chips is not constant but may vary depending on the workload. This capability is known under various brand names, such as Turbo Boost (Intel), Turbo Core (AMD), or Precision Boost (AMD). It is simply called "turbo" in this paper. In general, two clock frequencies are advertised for a processor: the base frequency and the maximum turbo frequency. The base frequency is the frequency one is expected to experience when a compute-bound workload using all the cores runs for a long time. The maximum turbo frequency is higher than the base frequency and is the frequency one *might* obtain under certain conditions, in particular if only a subset of the cores is active or if cores work intermittently.

One cannot fully understand the performance of modern multicores without understanding turbo. For instance, open-source microarchitecture simulators generally assume a fixed clock frequency, which may lead to inaccurate conclusions when evaluating certain hardware mechanisms whose behavior depends on memory latency, such as prefetchers. Unfortunately, the documentation for commercial chips rarely provides more than a superficial description of how turbo works. This paper highlights certain aspects of turbo that are not well known and proposes an "open-source" turbo, extrapolating from the scarce and sparse information disclosed so far.

Outside the industry, turbo is generally confounded with dynamic thermal management (DTM) [1]. However, there is an implicit performance contract in turbo that distinguishes it from DTM. Most DTM studies assume that chip temperature is known from integrated thermal sensors. Adjusting clock frequency and voltage depending on thermal sensors implies that frequency decreases when ambient temperature increases.

This raises the question of how to advertise base frequency when it depends on ambient temperature. Base frequency is an important selling point for a processor. While far from being the sole parameter determining processor performance, clock frequency is indeed a major one and, importantly, one that users understand. If the advertised base frequency is optimistic, say frequency at 20 °C (68 °F) ambient, some users will complain that they do not get the frequency they paid for. If the advertised base frequency is pessimistic, say frequency at 35 °C (95 °F) ambient, some users will notice that the processor is faster than advertised and will want an explanation (why would they buy a faster, more expensive processor?). Another problem is that of applications with real-time constraints. Making frequency depend on ambient temperature makes it more difficult for users to measure worst-case performance.

A possible solution is to define a *nominal ambient*

temperature T_{amb} and to implement a turbo such that clock frequency is practically independent of ambient temperature when below T_{amb}. This solution is called *deterministic* turbo in this paper. The chip manufacturer or the computer manufacturer have the freedom to configure the firmware and set T_{amb} to a suitable value. A high base frequency can be obtained by setting T_{amb} to a medium value. However, in this case, the base frequency is not a guaranteed frequency, and the meaning of T_{amb} must be explained to customers. Making the base frequency a guaranteed frequency requires to set T_{amb} to a high value. As for the maximum turbo frequency, it is not limited by temperature but by other constraints such as power delivery [2].

Intel's Turbo Boost, as implemented in Sandy Bridge chips, is a deterministic turbo in the sense defined above [3]. Clock frequency is determined not from thermal sensors but from an exponentially weighted moving average (EWMA) of the chip's power [4, 3]. Sandy Bridge chips still feature integrated thermal sensors monitoring circuit temperature (aka junction temperature) [4]. Sensors are here for DTM, i.e., for preventing circuit temperature from exceeding the maximum allowed junction temperature T_{max}. Sensor-based power throttling triggers only sporadically or under exceptional conditions such as atypical workload or ambient temperature exceeding T_{amb}.

While turbo exists mainly because of thermal constraints, a deterministic turbo is not a DTM mechanism, and does not replace one. The job of a deterministic turbo is not to prevent excursions above the temperature limit but to maximize clock frequency while keeping sensor-based power throttling rare. Deterministic turbo adjusts clock frequency not from the actual temperature but from a thermal model. The Sandy Bridge's EWMA is such thermal model. AMD's Turbo Core, too, uses a thermal model and is deterministic [5, 6, 7].

This paper proposes and describes a deterministic turbo. The goal is not to reverse-engineer what is implemented in existing commercial processors but to provide the outline of a possible "open-source" turbo. The rest of the paper consists of two main sections. Section 2 proposes and describes a thermal model for deterministic turbo. Then, based on this thermal model, Section 3 proposes and describes a deterministic turbo for multicores chips.

2 A simple thermal model

The thermal model is the keystone of a deterministic turbo. Ideally, a thermal model for deterministic turbo should be as accurate as possible when the actual ambient temperature equals the nominal value T_{amb}, so that clock frequency is not restrained more than what is sufficient for performance determinism, while keeping sensor-based power throttling rare. However, model accuracy and model simplicity are conflicting goals. So in practice, we need a thermal model that is reasonably accurate and still simple enough to provide calculated temperature in real time.

The model proposed in this paper is a *Foster RC ladder*, a type of compact thermal model [8]. AMD's Turbo Core uses an RC ladder [5, 6], probably a Foster-type one [9]. This section describes the proposed model. More details can be found in a companion technical report [10].

2.1 Single-input single-output model

The proposed model is based on linear system theory [11]. Linearity (aka superposition principle) is a customary and reasonably accurate approximation for thermal models of processor chips [8, 12].

Temperature $T(\boldsymbol{r}, t)$ in the thermal system is a function of location $\boldsymbol{r}$ and time t. Uniform initial temperature $T(\boldsymbol{r}, 0) = T_{amb}$ is assumed. Let

$$\mathcal{T}[g] := T - T_{amb}$$

denote the temperature rise over ambient air generated by power density $g(\boldsymbol{r}, t)$ (watts dissipated per unit volume). A single-input single-output (SISO) linear time invariant (LTI) system can be defined by assuming a power density of the form

$$g(\boldsymbol{r}, t) = m(\boldsymbol{r})p(t) \tag{1}$$

where $p(t)$, the total power dissipated in the thermal system, is the input of the SISO model. The power density map $m(\boldsymbol{r})$ is such that $\int m(\boldsymbol{r})d^3\boldsymbol{r} = 1$. The output of the SISO system is the relative temperature θ at a hot spot $\boldsymbol{r_\theta}$:

$$\theta(t) := \mathcal{T}[g](\boldsymbol{r_\theta}, t)$$

Point $\boldsymbol{r_\theta}$ is typically at a central position in a region of high average power density.

A SISO LTI system is completely characterized by its *step response* [11]. Here, the step response $H(t)$ is the temperature generated by a unit step power:

$$H(t) := \mathcal{T}[m(\boldsymbol{r})U(t)](\boldsymbol{r_\theta}, t)$$

where $U(t) = 0$ for $t < 0$, $U(t) = 1$ for $t \geq 0$. The step response can be obtained in the laboratory by measurements on a prototype platform or with a detailed thermal model. Note that $H(t)$ depends on the power density map $m(\boldsymbol{r})$, hence on the workload. In practice, several step responses should be collected corresponding to various power density maps and hot spot locations, from which an abstract worst-case step response can be defined [10].

From LTI system theory, the output is the convolution of the input and the derivative of the step response [11]:

$$\theta = p * H' = \int_0^t p(x)H'(t-x)dx \tag{2}$$

However, the convolution above is not a practical way to calculate temperature in real time, as its complexity increases linearly with time. Foster RC ladders are based on the observation that if the step responses of two different LTI systems are close to each other, these two systems produce approximately the same output when fed with the same input. More precisely, the idea is to approximate $H(t)$ as a weighted sum of exponentials:

$$H(t) = \sum_{i=1}^{n} H_i(1 - e^{-\lambda_i t}) \tag{3}$$

with finite n and $\lambda_i > 0$. As $\boldsymbol{r_\theta}$ is a hot spot, the fitting problem can be augmented with the constraint that the H_i's be nonnegative [10]. Functions of the form (3) with $\lambda_i > 0$ and $H_i \geq 0$ are called *Bernstein* functions. The details of the fitting procedure are described in the technical report [10]. In short, n and the λ_i's are chosen a priori, then the H_i's are obtained by solving a nonnegative least-squares (NNLS) problem.

A step response of the form (3) is interesting because it leads to a simple state-space model. The state-space model proposed in this paper is different from the one that is commonly used for Foster RC ladders (its advantage is discussed in Section 2.2). Let us define the n-dimensional state vector $\boldsymbol{X} = (X_1, \ldots, X_n)$ as follows:

$$X_i := p * (\lambda_i e^{-\lambda_i t}) = \lambda_i e^{-\lambda_i t} \int_0^t p(x)e^{\lambda_i x}dx$$

We have

$$\theta = \sum_{i=1}^{n} H_i X_i = \boldsymbol{H}^T \boldsymbol{X} \tag{4}$$

where $\boldsymbol{X}$ and $\boldsymbol{H}$ are column vectors. It can be verified that the following differential equation holds for all $i \in \{1, \ldots, n\}$:

$$X_i' = \lambda_i(p - X_i) \tag{5}$$

In the case when p is constant during $[t, t+\tau)$, solving (5) yields

$$X_i(t+\tau) = X_i(t)e^{-\lambda_i \tau} + p(1 - e^{-\lambda_i \tau}) \tag{6}$$

Note that (6) acts like an exponentially weighted moving average (EWMA). The degenerate case $n = 1$ corresponds to a single-stage RC ladder. In this case, θ is proportional to X_1, which explains the EWMA used to control the turbo clock frequency in the Intel Sandy Bridge [4, 3]. However, while a single RC stage might be sufficient for modeling and controlling the heatsink temperature [2], multiple RC stages are needed to model junction temperature accurately.

2.2 Multi-input multi-output model

The SISO model considers a single time-varying power source. Accurately modeling the temperature of a multicore chip requires considering multiple power sources. This section proposes a multi-input multi-output (MIMO) model derived from the SISO model. The MIMO model assumes multiple independent power sources as inputs:

$$g(\boldsymbol{r}, t) = \sum_{j=1}^{N} m_j(\boldsymbol{r})p_j(t) \tag{7}$$

where N is the number of sources and p_j is the power dissipated by the j^{th} source. What is considered an independent power source depends on how accurate we want the model to be. A source may be a core, a group of cores, a cache, a group of caches, etc. From the superposition principle, an N-input N-output MIMO model can be obtained by combining N^2 SISO systems. The step response $H_{ij}(t)$ is the temperature rise in source i when applying a one-watt power step in source j *only*:

$$H_{ij}(t) := \mathcal{T}[m_j(\boldsymbol{r})U(t)](\boldsymbol{r_{\theta_i}}, t)$$

where $\boldsymbol{r_{\theta_i}}$ is the point in source i where temperature is being monitored. Each of the N self-heating step responses $H_{ii}(t)$ is approximated as a Bernstein function. Each cross-heating step response $H_{ij}(t)$, $i \neq j$, is approximated as the difference between two Bernstein functions, for $m_j = (m_i + m_j) - m_i$.

The advantage of the state vector introduced in Section 2.1 is that, by selecting a common n and a common set of λ_i's for all N^2 step responses (which the NNLS fitting method facilitates), each source j is associated with a single state vector $\boldsymbol{X_j}$. That is, only N state vectors are needed, instead of N^2.

In the case when p_j is constant during $[t, t+\tau)$, (6) applies, which can be written in matrix notation:

$$\boldsymbol{X_j}(t+\tau) = e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}\boldsymbol{X_j}(t) + p_j[\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}]\mathbf{1} \quad (8)$$

where $\mathbf{D}[\boldsymbol{\lambda}]$ is the $n \times n$ diagonal matrix with the λ_i's on the diagonal, $\mathbf{I}$ is the $n \times n$ identity matrix and $\mathbf{1} = (1, \ldots, 1)$ is an n-dimensional column vector. From (4) and the superposition principle, relative temperature θ_i in source i is

$$\theta_i = \sum_{j=1}^{N} \boldsymbol{H_{ij}}^T \boldsymbol{X_j} \quad (9)$$

When constant powers p_j are applied for a sufficiently long time, temperatures reach a steady state. In steady state, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)$ is obtained from $\boldsymbol{p} = (p_1, \ldots, p_N)$ as

$$\boldsymbol{\theta} = \mathbf{R}\boldsymbol{p} \quad (10)$$

where $\mathbf{R}$ is the $N \times N$ matrix whose element in the i^{th} row and j^{th} column is $[\mathbf{R}]_{ij} = H_{ij}(\infty) = \boldsymbol{H_{ij}}^T\mathbf{1}$. Matrix $\mathbf{R}$ is positive definite, hence invertible [10].

3 A deterministic turbo for multicore chips

Several different turbo methods can be defined from the thermal model of Section 2. For instance, one may use the model like virtual thermal sensors and implement a turbo similar to sensor-based DTM. However, a thermal model offers possibilities beyond what real thermal sensors allow. This section proposes a turbo highlighting and exploiting these possibilities.

We consider a multicore chip where each core can adjust its clock frequency independently of other cores. The cores need not be identical, i.e., the multicore can be heterogeneous. The proposed turbo considers *junction* temperature only.[1] In this example, each core is modeled as an independent power source. To simplify the discussion, the uncore is treated like a core.

[1]Controlling enclosure skin temperature is important for certain handheld devices [2].

The goal of the turbo is to maximize clock frequency under the constraint $T_{amb} + \theta \leq T_{max}$, where T_{max} is the maximum allowed junction temperature and T_{amb} is the nominal ambient temperature.

3.1 Steady-state power assignment

A sensor-based DTM would try to maximize multicore performance by running each core at the maximum allowed power such that the temperature limit is not violated, e.g., by reducing the clock frequency of a core when temperature in that core approaches T_{max}. When all cores are running a compute-bound workload, such control strategy results in the temperature of each core being close to T_{max}. This means that the steady-state power of each core is completely determined, as $\boldsymbol{p} = \mathbf{R}^{-1}\boldsymbol{\theta}$ from equation (10). However, this may not be what we want.

For example, consider parallel threads executing on 9 identical cores laid out as a 3x3 grid. Without a thermal limit, the core at the center of the grid would end up being hotter than the 8 other cores (the border cores) just because it is surrounded by them. With sensor-based DTM, the center core reduces its power (hence its clock frequency) more than the border cores in order not to exceed the temperature limit. Consequently, the thread on the center core is slower than the other threads and reaches the synchronization point later, slowing the execution.

The thermal asymmetry could be dealt with by forcing the clock frequency to be equal on all cores. However, this would reduce multicore throughput when several independent tasks with different characteristics run simultaneously [13] (one task hitting the temperature limit slows all the other tasks).

Instead, the solution proposed in this paper, called *steady-state power assigment* (SSPA), is to let the firmware control the power that each core will dissipate when temperatures reach a steady state. Let us denote $\boldsymbol{p^*} = (p_1^*, \ldots, p_N^*)$ the steady-state power values assigned by the firmware to the cores. Vector $\boldsymbol{p^*}$ is redefined when the set of active cores changes. An *inactive* core is a core that has been in a sleep state for a long time and is likely to remain so for some time. The p_i^* of an inactive core is the standby power corresponding to the core's sleep state.

The p_i^*'s of active cores are set under the following constraint. Let $\boldsymbol{\theta^*} = (\theta_1^*, \ldots, \theta_N^*)$ be the vector of steady-state temperatures corresponding to $\boldsymbol{p^*}$, that is, $\boldsymbol{\theta^*} = \mathbf{R}\boldsymbol{p^*}$. The firmware sets the p_i^*'s of active cores such that

$$T_{amb} + \max\{\theta_1^*, \ldots, \theta_N^*\} \leq T_{max}$$

SSPA is defined as follows:

SSPA. *If $\boldsymbol{p^*}$ remains unchanged for a time long enough for a thermal steady state to settle and if power values p_i^* are attainable, the steady-state $\boldsymbol{p}$ must be equal or close to $\boldsymbol{p^*}$.*

Two important points must be noted. First, SSPA does not say how $\boldsymbol{p^*}$ should be set. For instance, identical active cores might be assigned an equal p_i^* in all situations, or different p_i^*'s depending on workload characteristics [14]. This question is beyond the scope of this paper. Second, SSPA does not dictate that actual power $\boldsymbol{p}$ be equal to $\boldsymbol{p^*}$ at all times, but only that it *converge* towards $\boldsymbol{p^*}$. This degree of freedom allows to exploit thermal transients by boosting the power of a cold core temporarily above its assigned p_i^*.

A simple way to enforce SSPA while taking advantage of thermal transients is to make $\boldsymbol{\theta}$ converge towards $\boldsymbol{\theta^*}$, which forces power $\boldsymbol{p}$ to converge towards $\boldsymbol{p^*}$. To the best of the author's knowledge, SSPA is an original proposition.

3.2 An SSPA-compatible turbo

The implicit thermal model behind sensor-based DTM is simply that temperature decreases when power is reduced sufficiently. Sensor-based DTM cannot determine in advance what power value will keep temperature constant. In contrast, thanks to its thermal model, deterministic turbo knows how temperature relates to power, so it can turn a temperature control problem into a power control problem. Power control can react quickly to workload behavior changes (e.g., every few tens of microseconds [15]). Power control is a relatively mature topic and will not be described here (see, e.g., [14, 15]).

The proposed turbo uses a fixed timestep τ. That is, $\boldsymbol{\theta}$ and the $\boldsymbol{X_j}$'s are updated every τ. At time t, the turbo algorithm calculates for each core i a power quota q_i. Power quotas $\boldsymbol{q} = (q_1, \ldots, q_N)$ are the targets for power control during $[t, t+\tau)$. That is, power control adjusts clock frequency and voltage, possibly multiple times during the interval, so that the core power p_i is as close to q_i as possible.

The state $\boldsymbol{X_i}$ of each core i is updated at time $t+\tau$ with the power p_i dissipated by the core, according to (8). Power quotas are calculated as follows. Assuming constant p_i's during $[t, t+\tau)$, we get from (9) and (8)

$$\theta_i(t+\tau) = \theta_i(t) - \Delta_i(t,\tau) + \sum_{j=1}^{N} H_{ij}(\tau) p_j \qquad (11)$$

condition on core i	$[\mathbf{B}]_{ij}$	Y_i
inactive	δ_{ij}	standby power p_i^*
$\theta_i(t) < \theta_i^* - \eta$ and $\hat{\theta}_i(t+\tau) < \theta_i^* - \eta$	δ_{ij}	maximum core power π_i
$\theta_i(t) < \theta_i^* - \eta$ and $\hat{\theta}_i(t+\tau) \geq \theta_i^* - \eta$	$H_{ij}(\tau)$	$\theta_i^* - \theta_i(t) + \Delta_i(t,\tau)$
$\theta_i^* - \eta \leq \theta_i(t) \leq \theta_i^*$	$H_{ij}(\tau)$	$\theta_i^* - \theta_i(t) + \Delta_i(t,\tau)$
$\theta_i(t) > \theta_i^*$	$H_{ij}(\tau)$	$\epsilon[\theta_i^* - \theta_i(t)] + \Delta_i(t,\tau)$

Table 1: *The power quotas q_i are obtained by solving the system of linear equations $\mathbf{Bq} = \boldsymbol{Y}$. Notation δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ for $i = j$, $\delta_{ij} = 0$ for $i \neq j$). Parameters η and ϵ are positive.*

with

$$\Delta_i(t,\tau) := \sum_{j=1}^{N} \boldsymbol{H_{ij}}^T [\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}] \boldsymbol{X_j}(t) \qquad (12)$$

Note that when τ is less than the time for heat to diffuse from a core to another core, we have $|H_{ij}(\tau)| \ll H_{ii}(\tau)$ for $i \neq j$, and an approximate upper bound for $\theta_i(t+\tau)$ can be obtained as

$$\hat{\theta}_i(t+\tau) := \theta_i(t) - \Delta_i(t,\tau) + H_{ii}(\tau)\pi_i$$

where π_i is an upper bound for core i's power at maximum clock frequency. If core i is predicted to stay below θ_i^* during $[t, t+\tau)$, then $q_i = \pi_i$. Otherwise, q_i is obtained by setting $\theta_i(t+\tau) = \theta_i^*$ in (11). Precisely, the power quotas $\boldsymbol{q}$ are obtained by solving the system of linear equations

$$\mathbf{Bq} = \boldsymbol{Y} \qquad (13)$$

where the $N \times N$ matrix $\mathbf{B}$ and the vector $\boldsymbol{Y} = (Y_1, \ldots, Y_N)$ are set as shown in Table 1. Parameter $\eta > 0$ is for minimizing overshoot (as $\hat{\theta}_i(t+\tau)$ is only an approximate upper bound).

The case $\theta_i(t) > \theta_i^*$ (last row of Table 1) can happen after θ_i^* has been lowered. In this case we force θ_i to decrease, which is sufficient to enforce SSPA [10]. This is done by setting Y_i so that θ_i decays exponentially towards θ_i^*. Parameter ϵ defines the time constant τ/ϵ of the exponential decay.

Calculations for the turbo algorithm can be offloaded to a firmware thread running on a CPU core. The CPU core is interrupted once per timestep to execute the firmware thread. The core might be customized to minimize the interrupt penalty (the firmware thread does not use architectural registers).

The algorithm's data is stored in one or two dedicated 4 KB physical pages, depending on N (for large N values, say $N = 20$, the data footprint is dominated by the $H_{ij}(\tau)$ constants). Persistent data can be prefetched into the caches before interrupting the CPU core. Calculations for one timestep typically do not exceed a few microseconds [10], which is a tiny fraction of the CPU core time, considering a timestep τ of a few milliseconds.

4 Conclusion

The proposed deterministic turbo possesses two interesting qualities. First, it allows the firmware to control the steady-state power of each core while allowing cold cores to boost their power dissipation temporarily above the assigned steady-state value. Second, it turns a temperature control problem into a power control problem, allowing clock frequency to adapt to fast-changing workload behavior.

Implementing the proposed deterministic turbo, or one inspired from it, in a microarchitecture simulator should be relatively straightforward. Thermal step responses can be obtained with open-source software tools for modeling processor power and temperature [16, 17]. It should be emphasized that, to simulate the effect of deterministic turbo on clock frequency, one need not simulate true temperature but *only the thermal model used by the turbo, including the model's inaccuracies.*. This should not slow simulations in any significant way.

References

[1] K. Kong, S. W. Chung, and K. Skadron. Recent thermal management techniques for microprocessors. *ACM Computing Surveys*, 44(3), June 2012.

[2] E. Rotem, R. Ginosar, A. Mendelson, and U. C. Weiser. Power and thermal constraints of modern system-on-a-chip computer. *Microelectronics Journal*, 46(12), December 2015.

[3] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann. Power-management architecture of the Intel microarchitecture code-named Sandy Bridge. *IEEE Micro*, 32(2), March 2012.

[4] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann. Power management of the 2nd generation Intel Core microarchitecture, formerly code-named Sandy Bridge. In *Hot Chips*, 2011. `http://www.hotchips.org/archives`.

[5] S. Nussbaum. AMD Trinity APU. In *Hot Chips*, 2012. `http://www.hotchips.org/archives`.

[6] P. Dongara, L. Bircher, and J. Darilek. AMD Richland client APU. In *Hot Chips*, 2013. `http://www.hotchips.org/archives`.

[7] AMD. Advanced power management helps bring improved performance to highly integrated x86 processors. white paper, 2014.

[8] M.-N. Sabry. Dynamic compact thermal models used for electronic design: a review of recent progress. In *ASME International Electronic Packaging Technical Conference and Exhibition*, 2003.

[9] Y. Yang, S. Sundaram, G. Refai-Ahmed, and M. Touzelbaev. Fast prediction of temperature evolution in electronic devices for run-time thermal management applications. In *ASME International Mechanical Engineering Congress and Exposition (IMECE)*, 2010.

[10] P. Michaud. A simple model of processor temperature for deterministic turbo clock frequency. research report RR-9308, Inria, December 2019. `https://hal.inria.fr/hal-02391970`.

[11] C. L. Phillips, J. M. Parr, and E. A. Riskin. *Signals, systems, and transforms*. Prentice Hall, fourth edition, 2008.

[12] S.-L. Kuo, C.-W. Pan, P.-Y. Huang, C.-T. Fang, S.-Y. Hsiau, and T.-Y. Chen. An innovative heterogeneous SoC thermal model for smartphone systems. In *Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, 2018.

[13] J. Donald and M. Martonosi. Techniques for multicore thermal management: classification and new exploration. In *International Symposium on Computer Architecture (ISCA)*, 2006.

[14] A. K. Mishra, S. Srikantaiah, M. Kandemir, and C. R. Das. CPM in CMPs: coordinated power management in chip-multiprocessors. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2010.

[15] M. Floyd, M. Ware, K. Rajamani, T. Gloekler, B. Brock, P. Bose, A. Buyuktosunoglu, J. C. Rubio, B. Schubert, B. Spruth, J. A. Tierno, and L. Pesantez. Adaptive energy-management features of the IBM POWER7 chip. *IBM Journal of Research and Development*, 55(3), May 2011.

[16] A. Roelke, R. Zhang, K. Mazumdar, K. Wang, K. Skadron, and M. R. Stan. Pre-RTL voltage and power optimization for low-cost, thermally challenged multicore chips. In *International Conference on Computer Design (ICCD)*, 2017.

[17] H. Sultan, A. Chauhan, and S. R. Sarangi. A survey of chip-level thermal simulators. *ACM Computing Surveys*, 52(2), May 2019.