A Simple Model of Processor Temperature for Deterministic Turbo Clock Frequency

Pierre Michaud

A Simple Model of Processor Temperature for Deterministic Turbo Clock Frequency

Pierre Michaud

RESEARCH REPORT
N° 9308
December 2019
Project-Team PACAP

ISSN 0249-6399 ISRN INRIA/RR--9308--FR+ENG

A Simple Model of Processor Temperature for Deterministic Turbo Clock Frequency

Pierre Michaud

Project-Team PACAP

Research Report n° 9308 — version 2 — initial version December 2019 — revised version October 2022 — 43 pages

Abstract: As power dissipation and circuit temperature constrain their performance, modern processors feature turbo control mechanisms to adjust the voltage and clock frequency dynamically so that circuit temperature stays below a certain limit. In particular, turbo control exploits the fact that, after a long period of low processor activity, the thermal capacity of the chip, its package and the heatsink can absorb heat at a relatively fast rate during a certain time, before the temperature limit constrains that rate. Hence power dissipation can be temporarily boosted above the average sustainable value. The turbo control must monitor circuit temperature continuously to maximize the clock frequency. Temperature can be monitored by reading the integrated thermal sensors. However, making the clock frequency depend on thermal sensor readings implies that processor performance depends on ambient temperature. Yet this form of performance non-determinism is a problem for certain processor makers. A possible solution is to determine the clock frequency not from the true temperature but from a thermal model based on the nominal ambient temperature. Such model should be as accurate as possible in order to prevent sensor-based protection from triggering but sporadically, without hurting performance by overestimating temperature too much. The model should also be simple enough to provide calculated temperature in real time. This document proposes such thermal model and a turbo control based on that model.

Key-words: multicore processor, junction temperature, transient, clock frequency, power density, compact thermal model, step response

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Un modèle simple de température des processeurs pour une fréquence d'horloge turbo déterministe

Résumé : La performance des processeurs modernes étant contrainte par la consommation électrique et la température des circuits, ceux-ci comportent des mécanismes de contrôle turbo dont la fonction est de régler la tension électrique et la fréquence d'horloge afin de maintenir à tout instant la température des circuits sous la limite. En particulier, le contrôle turbo exploite le fait qu'après une longue période de faible activité du processeur, la capacité thermique de la puce, de son boitier et du radiateur peut absorber la chaleur à un taux relativement élevé pendant un certain temps, avant que la limite en température ne restreigne ce taux. Ainsi la consommation électrique peut temporairement dépasser la valeur moyenne que la puce peut dissiper sur une période prolongée. Le contrôle turbo doit évaluer la température constamment afin de maximiser la fréquence d'horloge. La température peut être obtenue en lisant les capteurs intégrés sur la puce. Cependant, rendre la fréquence d'horloge dépendante des capteurs implique que la performance du processeur dépend de la température ambiante. Or cette forme de non-déterminisme de la performance est un problème pour certains fabricants de processeurs. Une solution possible est de déterminer la fréquence d'horloge non pas à partir de la température réelle mais à partir d'un modèle de température basé sur la température ambiante nominale. Un tel modèle doit être aussi précis que possible afin d'empêcher les capteurs intégrés d'enclencher la protection thermique sauf de manière occasionnelle, tout en évitant de nuire à la performance par une surestimation excessive de la température. Le modèle doit aussi être suffisamment simple pour fournir une température calculée en temps réel. Ce document propose un modèle de température répondant à ces critères, et un contrôle turbo basé sur ce modèle.

Mots-clés : processeur multi-coeur, température de jonction, régime transitoire, fréquence d'horloge, densité de puissance, modèle thermique compact, réponse indicielle

1 Introduction

Active integrated circuits age faster at high temperatures, because of various phenomena such as electromigration and gate oxide wearout [82]. Also, transistor leakage hence static power consumption increases with temperature. Moreover, high temperatures make transistors and wires slower. Therefore, microprocessors are engineered so that circuit temperature, aka *junction* temperature, does not exceed a maximum value T_{max}, e.g., 100 °C.

A possible way to prevent temperature from exceeding T_{max} is to design the processor conservatively so that thermal violations never happen. However, this approach sacrifices processor performance. Instead, modern processors feature thermal protection mechanisms, aka dynamic thermal management (DTM), consisting of integrated thermal sensors and methods for throttling power dissipation when junction temperature approaches T_{max} [69, 61, 12, 73]. DTM allows higher processor performance in common situations, with thermal protection triggering only for atypical workloads or under exceptional conditions such as dysfunctional fan or ambient temperature above the nominal value.

However, the need for high performance has pushed processor makers to exploit temporal power dissipation variations. Power dissipation may fluctuate for various reasons:

- Microarchitectural activity depends on applications' dynamic behavior, which is variable [34].
- Applications have variable thread-level parallelism [10].
- Interactive applications typically generate short bursts of high processor activity when responding to user inputs, interspersed with phases of low activity while waiting for user inputs [34, 10].
- Latency-critical servers are underutilized (i.e., overprovisioned) in order to keep queuing delays short [47, 37]. Processor activity fluctuates because of this.

During periods of low processor activity, power dissipation and junction temperature are low. When a period of high activity starts after a long period of low activity, junction temperature increases, yet not instantaneously owing to the heat capacity of the chip, the package and the heatsink. There is a certain time during which the chip can dissipate a power exceeding the TDP (thermal design power)[1] while keeping its junction temperature below, or at T_{max} [14, 65]. Today's processors adjust their clock frequency dynamically and automatically to exploit such thermal transients. This capability is advertised under various brand names, such as Intel's Turbo Boost, AMD's Turbo Core, AMD's Precision Boost, etc. [65, 62, 24, 13]

The details of such *turbo* control, implemented in the processor firmware, are generally not disclosed. In particular, whether the turbo control uses thermal sensors or not is not stated explicitly, in general. For instance, it seems that the Intel Sandy Bridge turbo does not use thermal sensors. The Sandy Bridge still features thermal sensors for thermal protection, but uses an EWMA (exponentially weighted moving average) algorithm for determining the turbo clock frequency [65]. Rotem et al. gave a hint [65]:

> "The architectural power predictor provides a consistent turbo behavior while minimizing the die-to-die variations and dependency on ambient temperature."

Emurian et al. gave another hint, from performance measurements they did on an Intel Haswell chip while varying the fan speed [25]:

[1]TDP is the usual term for the power that the chip can dissipate in a *sustained* manner without exceeding T_{max}, under normal operating conditions.

> "Despite being motivated primarily by thermal concerns, we found that the Turbo Boost control policy implemented by this Intel chip does not depend directly on temperature."

A couple of Intel patents mention that some equipment manufacturers want *deterministic* performance [55, 66]. Here, "deterministic" means that supposedly identical chips should provide the same performance, and that performance should not depend on external factors such as ambient temperature. This implies that sensor-based thermal protection should trigger only sporadically when the ambient temperature is below the nominal value stipulated by the processor maker. This also implies that the turbo algorithm cannot use the integrated thermal sensors, as the firmware does not know the actual ambient temperature.

Clearly, there is a contradiction between maximizing performance and making it deterministic. Some users do not care about performance determinism and prefer that the processor accelerate opportunistically whenever the ambient temperature is less than the nominal value. In fact, it seems that AMD's Turbo Core uses thermal sensor information, as observed by Lo and Kozyrakis [46]:

> "At the hardware level, there are significant differences between AMD's and Intel's TurboMode controller. AMD's implementation reacts to current and temperature sensors on the CPU die to adjust the frequency boost. [...] Theoretically, Intel's implementation has a more repeatable TurboMode boost, while AMD's implementation is more sensitive to external factors such as ambient temperature."

Nevertheless, performance determinism matters to AMD too. Its Turbo Core technology mitigates performance non-determinism by introducing a calculated temperature in the control loop [1]. As for the more recent Precision Boost technology, AMD chips provide the possibility to configure the firmware and choose between power determinism and performance determinism [2].

Implementing a deterministic turbo clock frequency requires a thermal model based on certain pessimistic assumptions such as assuming that the ambient temperature is equal to the nominal value. A thermal model for deterministic turbo clock frequency should possess the following qualities:

- Under normal conditions, in particular when the ambient temperature is less than the nominal value, it should prevent thermal protection to trigger but sporadically.
- It should not hurt performance by overestimating temperature too much.
- It should be simple enough to provide calculated temperature in real time.

The first and second point, together, mean that the model should be as accurate as possible when the actual ambient temperature is equal to the nominal value. However, model accuracy and model simplicity are conflicting goals, in general. A thermal model for deterministic turbo should be reasonably accurate without being too complex.

The goal of this document is not to reverse-engineer the turbo algorithms implemented in current commercial processors but to propose a thermal model possessing the above-mentioned qualities and a turbo algorithm based on that model.

This document is organized as follows. Section 2 describes the proposed thermal model in its basic form, which is single-input single-output (SISO). The model is based on the thermal step response, which must be obtained by measurement on a real chip or by simulation with a detailed thermal model. Section 3 shows that, for the SISO model, if power dissipation is an affine function of clock frequency and temperature, then maximizing performance amounts to maximizing temperature. A turbo control based on the SISO thermal model is proposed in

Section 4. Section 5 introduces the idea that the thermal step response may not be a real step response but an abstract one obtained by combining several step responses. Section 6 proposes a multi-input multi-output (MIMO) thermal model directly derived from the SISO model. A turbo control based on the MIMO thermal model is proposed in Section 7. The idea of *steady-state power assignment* is introduced, such that the firmware can control the steady-state power allotted to each core while letting the turbo control exploit thermal transients. Section 8 presents some high-level simulations of the proposed MIMO turbo. Some related works are mentioned in Section 9. Finally, Section 10 concludes this document.

The main mathematical notations are listed in Table 1.

2 A thermal model for transient temperature

This section describes the proposed thermal model, which is based on several simplifying assumptions.

2.1 Heat equation

The first assumption is that temperature in the chip, the package and the heatsink can be modeled with the classical heat conduction equation [15]:

$$\boldsymbol{\nabla} \cdot \kappa \boldsymbol{\nabla} T + g = c_v \frac{\partial T}{\partial t} \tag{1}$$

where temperature T and volume power density g are functions of the 3-dimensional location $\boldsymbol{r}$ and of time t. Thermal conductivity κ and volumetric heat capacity c_v both depend on $\boldsymbol{r}$ for a heterogeneous system consisting of different isotropic materials (silicon, copper, etc.). Equation (1) is based on Fourier's law, which is an accurate model of heat conduction for silicon dies as thin as a few micrometers [40].

To provide definite temperatures, equation (1) must be completed with prescribed boundary conditions over a closed surface $\mathcal{S}$ and initial temperatures in the volume $\mathcal{V}$ enclosed by $\mathcal{S}$. Surface $\mathcal{S}$ contains the interface between the heatsink and the ambient air.

2.2 Superposition principle

The second assumption is that the superposition principle is valid, that is, temperature depends linearly on power density. This implies the following.

First, equation (1) can be linearized by assuming that thermal conductivity κ and heat capacity c_v are independent of temperature. It should be noted that, for the silicon die, this assumption is not quite exact, as silicon thermal conductivity varies from 154 W/(mK) to 112 W/(mK) when temperature varies from 20 °C to 100 °C (see equation (23) in [40]). Assuming that heat capacity is independent of temperature implies that there is no phase change material in the package or in the heatsink.

Second, linear boundary condition are assumed over the enclosing surface $\mathcal{S}$:

$$-\kappa \boldsymbol{\nabla} T \cdot \boldsymbol{n} = A \times (T - T_{amb}) \tag{2}$$

where $\boldsymbol{n}$ is the position-dependent unit vector perpendicular to $\mathcal{S}$ and directed towards the outside of $\mathcal{V}$. Equation (2) corresponds to Newton's law of cooling. The ambient temperature T_{amb} is assumed constant and uniform.

notation	meaning	remark
$\mathcal{V}$	chip + package + heatsink	volume in 3D space
$\mathcal{S}$	boundary of $\mathcal{V}$	closed surface
$\boldsymbol{r}$	point of $\mathcal{V}$	$\boldsymbol{r} = (x, y, z)$
T	temperature (function of $\boldsymbol{r}$ and t)	in K or in °C
T_{max}	maximum junction temperature	we want $T \leq T_{max}$
T_{amb}	ambient temperature	
t	time	$T = T_{amb}$ at $t = 0$
τ	shifted time or time step	
g	volume power density	in W/m^3
$\boldsymbol{\nabla}$	$\boldsymbol{\nabla} = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$	nabla aka del
u	relative temperature (function of $\boldsymbol{r}$ and t)	$u := T - T_{amb}$
κ	thermal conductivity	in W/(mK)
c_v	volumetric heat capacity	in J/(m^{3}K)
A	heat transfer coefficient	in W/(m^{2}K)
$\boldsymbol{n}$	outward unit normal vector to $\mathcal{S}$	
$\mathcal{T}[g]$	relative temperature generated by g	see Section 2.2
m	power density map (function of $\boldsymbol{r}$)	$\int_{\mathcal{V}} m(\boldsymbol{r}) d^3\boldsymbol{r} = 1$
p	total chip power (function of t)	in W
$\boldsymbol{r_\theta}$	point where temperature is being monitored	aka hot spot
θ	relative temperature at point $\boldsymbol{r_\theta}$	$\theta(t) := u(\boldsymbol{r_\theta}, t)$
θ_{max}	$\theta_{max} := T_{max} - T_{amb}$	see Section 4.1
f	normalized (dimensionless) clock frequency	$f \in [0, 1]$
v	normalized (dimensionless) voltage	$v \in [0, 1]$
U	unit step function	$U(t) = 0$ for $t < 0$ $U(t) = 1$ for $t \geq 0$
H	step response $H(t) := \mathcal{T}[m(\boldsymbol{r})U(t)](\boldsymbol{r_\theta}, t)$	see Section 2.4
h	impulse response $h = \frac{dH}{dt}$	
$*$	$p * h = \int_0^t p(x)h(t - x)dx$	convolution
n, λ_i, h_i, H_i	$h(t) \simeq \sum_{i=1}^n h_i e^{-\lambda_i t}, \quad H_i = \frac{h_i}{\lambda_i}$	see Section 2.5
X_i	$X_i = p * (\lambda_i e^{-\lambda_i t})$	see Section 2.6
$\boldsymbol{X}$	column vector $(X_1, \ldots, X_n)$	aka state vector
$\boldsymbol{H^T}$	row vector $(H_1, \ldots, H_n)$	transpose of $\boldsymbol{H}$
$\boldsymbol{\lambda}$	column vector $(\lambda_1, \ldots, \lambda_n)$	
$\boldsymbol{h}$	column vector $(h_1, \ldots, h_n)$	
$\mathbf{D}[\boldsymbol{\lambda}]$	$n \times n$ diagonal matrix with $\boldsymbol{\lambda}$ on the diagonal	
$\mathbf{I}$	$n \times n$ identity matrix	
$\mathbf{1}$	n-dimensional column vector $(1, \ldots, 1)$	
det	determinant of a square matrix	
$[\mathbf{Q}]_{ij}$	element in row i and column j of matrix $\mathbf{Q}$	
N	number of cores	see Section 6
p_j	power of core j	
θ_i	relative temperature at point $\boldsymbol{r_{\theta_i}}$ in core i	$\theta_i(t) := u(\boldsymbol{r_{\theta_i}}, t)$
$\boldsymbol{X_j}$	state vector of core j	n-dimensional vector
$\boldsymbol{p^*}$	allotted steady-state p_j's $\boldsymbol{p^*} = (p_1^*, \ldots, p_N^*)$	see Section 7
$\boldsymbol{\theta^*}$	allotted steady-state θ_i's $\boldsymbol{\theta^*} = (\theta_1^*, \ldots, \theta_N^*)$	see Section 7
$\mathbf{R}$	steady-state thermal resistance matrix	$\boldsymbol{\theta^*} = \mathbf{R}\boldsymbol{p^*}$

Table 1: Main mathematical notations. Vectors and matrices are in bold. Notations not listed here are often defined close to where they are used.

The heat transfer coefficient A depends on location $\boldsymbol{r}$ but is assumed constant in time[2] and independent of temperature.[3] As for the initial temperature, $T = T_{amb}$ is assumed everywhere in $\mathcal{V}$ at $t = 0$. Under these assumptions, which are commonly used for thermal modeling of integrated circuits, the boundary value problem is linear and the superposition principle applies. Defining the relative temperature

$$u := T - T_{amb},$$

let us denote $\mathcal{T}[g]$ the temperature $u(\boldsymbol{r}, t)$ generated by power density $g(\boldsymbol{r}, t)$. Then, the superposition principle means that for any power densities $g_1(\boldsymbol{r}, t)$ and $g_2(\boldsymbol{r}, t)$ and for any scalars a_1 and a_2,

$$\mathcal{T}[a_1 g_1 + a_2 g_2] = a_1 \mathcal{T}[g_1] + a_2 \mathcal{T}[g_2] \tag{3}$$

2.3 Single-input single-output (SISO) system

In this section, temperature is modeled as a single-input single-output (SISO) system, where the input is total chip power and the output is temperature at a particular chip location. The SISO model is based on the assumption that power density is of the form

$$g(\boldsymbol{r}, t) = m(\boldsymbol{r}) p(t) \tag{4}$$

where $m(\boldsymbol{r})$ is the power density map, with

$$\int_{\mathcal{V}} m(\boldsymbol{r}) d^3\boldsymbol{r} = 1$$

and where $p(t)$ is the total power, with $p(t) = 0$ for $t < 0$.

The output of the SISO system is temperature at a hot spot $\boldsymbol{r}_{\boldsymbol{\theta}}$, i.e., a point inside the heat source where power density is non-null, generally a central point in a region of high average power density. Relative temperature at that point is denoted θ:

$$\theta(t) := u(\boldsymbol{r}_{\boldsymbol{\theta}}, t) \tag{5}$$

A more general model for power density when the CPU chip is active is

$$g(\boldsymbol{r}, t) = m_s(\boldsymbol{r}, t) b(v, T) P_s(t) + m_d(\boldsymbol{r}, t) f v^2 P_d(t) \tag{6}$$

where f and v are the normalized (dimensionless) clock frequency and voltage, $P_s(t)$ is the static power at maximum voltage and temperature ($v = 1$, $T = T_{max}$), $P_d(t)$ is the dynamic power at maximum voltage and frequency ($v = 1$, $f = 1$), and $b(v, T)$ models the dependence of leakage power on voltage and temperature.

The following conditions are needed for (6) to match (4):

1. The static and dynamic power density maps must be identical:

$$m_s(\boldsymbol{r}, t) = m_d(\boldsymbol{r}, t)$$

2. The power density map must be independent of time:

$$m_d(\boldsymbol{r}, t) = m_d(\boldsymbol{r})$$

[2]If the heatsink features a fan, it is assumed that the fan rotation speed is kept constant.

[3]Newton's law of cooling is an approximation, particularly if the heatsink is cooled by natural convection or if heat transfer by thermal radiation cannot be neglected [56].

3. $b(v,T)$ must be independent of $\boldsymbol{r}$. Specifically, it is assumed that θ can be used as a proxy for T in modeling the dependence of leakage on temperature:

$$p = \tilde{b}(v, T_{amb} + \theta)P_s + fv^2 P_d \tag{7}$$

If the above conditions are not met, assumption (4) is inaccurate. Nevertheless, during periods of low processor activity, when θ is supposed to be low, it is not a problem if temperature is not modeled very accurately. What is important is that $p(t)$ be estimated with sufficient accuracy,[4] as explained in Appendix A. So (4) need be accurate only during periods of high processor activity.

2.4 Linear time-invariant system

From the superposition principle, time invariance and assumption (4), temperature θ can be obtained as a superposition integral [3, 54]:

$$\theta = p * h = \int_0^t p(x)h(t-x)dx \tag{8}$$

where $*$ denotes the convolution operation and where $h(t)$ is defined as follows. Let us apply a unit step power

$$g(\boldsymbol{r},t) = m(\boldsymbol{r})U(t)$$

with

$$U(t) := \begin{cases} 0 & \text{for } t < 0 \\ 1 & \text{for } t \geq 0 \end{cases}$$

and let $H(t)$ be the resulting temperature response at point $\boldsymbol{r_\theta}$:

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

Then $h(t)$ is the derivative of the step response $H(t)$:

$$h(t) := \frac{dH}{dt}$$

Convolution (8) defines a SISO, causal, linear time-invariant (LTI) system with power $p(t)$ as input, temperature $\theta(t)$ as output and $h(t)$ as *impulse response* [16, 54]. Quantity $H(\infty)$, that is, the steady-state temperature when dissipating 1 watt of power, is usually called *junction-to-ambient thermal resistance*. Note that it depends on the power density map $m(\boldsymbol{r})$.

2.5 Completely monotone impulse response

The impulse response $h(t)$ determines how $\theta(t)$ varies with $p(t)$. Characterizing $h(t)$ is important for understanding transient temperature on the chip.

Intuitively, the step response $H(t)$ is an increasing function of time, hence $h(t)$ is positive.[5] Moreover, point $\boldsymbol{r_\theta}$ is not a random point on the chip but a hot spot, located inside a region

[4]The dissipated energy can be estimated with a calibrated model taking as input various activity counters [34, 33].

[5]Sketch of a proof. Let $T(\boldsymbol{r},t) = \mathcal{T}[m(\boldsymbol{r})U(t)]$ and $T' = \frac{\partial T}{\partial t}$. From (1), and as $T(\boldsymbol{r},0) = T_{amb}$, we have $T'(\boldsymbol{r},0) = \frac{m(r)}{c_v}$. T' is the solution of the boundary value problem with no heat source, null ambient temperature, and non-negative initial temperature. Temperature at a local minimum is non-decreasing, hence $T' \geq 0$, and $h(t) = T'(\boldsymbol{r_\theta},t) \geq 0$.

of high power density. At such point, $H(t)$ is not only increasing but concave. That is, $h(t)$ is decreasing. Although I have no mathematical proof, I conjecture that this is always true or approximately true *at a hot spot*.[6] I make an even stronger claim that I call the *completely monotone impulse response* (CMIR) hypothesis:

> The thermal impulse response $h(t)$ at a hot spot is *completely monotone*, or close to a completely monotone function.

Complete monotonicity is defined as follows [49, 70]. A function $h(t)$ is completely monotone on $(0, +\infty)$ if, for all $t > 0$ and for all $j = 0, 1, 2, 3, \ldots,$

$$(-1)^j h^{(j)}(t) \geq 0$$

where $h^{(j)}$ is the j^{th} derivative. In particular, h is positive ($h^{(0)} \geq 0$), decreasing ($h^{(1)} \leq 0$), and convex ($h^{(2)} \geq 0$).

Bernstein's theorem states, loosely speaking, that completely monotone functions coincide with *positive* sums of decreasing exponentials [70]. More specifically, I conjecture that $h(t)$ can be approximated as

$$h(t) = \sum_{i=1}^{n} h_i e^{-\lambda_i t} \tag{9}$$

with

$$\lambda_i > 0 \tag{10a}$$

$$h_i \geq 0 \tag{10b}$$

Equation (9) is equivalent to

$$H(t) = \sum_{i=1}^{n} H_i (1 - e^{-\lambda_i t}) \tag{11}$$

where

$$H_i = \frac{h_i}{\lambda_i}$$

is non-negative. Non-negative functions whose first derivative is completely monotone are called *Bernstein functions* [70]. That is, the impulse response $h(t)$ is completely monotone iff the step response $H(t)$ is a Bernstein function. Hence the following statement is equivalent to the CMIR hypothesis:

> The thermal step response $H(t)$ at a hot spot is a Bernstein function, or close to a Bernstein function.

Under similar assumptions, Gerstenmaier and Wachutka proved that temperature at any point in $\mathcal{V}$ can be written as a discrete, infinite sum ($n = \infty$) of decreasing exponentials [26, 27]. That is, they proved (9) and (10a). So the CMIR hypothesis rests on proving (10b). Currently, I know no such proof.[7] Nevertheless, I have two arguments to support the CMIR hypothesis, a heuristic argument, and an empirical one.

The heuristic argument is based on Codecasa's work [19, 17]. Instead of focusing on temperature at a particular hot spot, as I do, Codecasa defines an average temperature weighted by the power density map:

$$\bar{u}(t) := \int_{\mathcal{V}} m(\boldsymbol{r}) u(\boldsymbol{r}, t) d^3\boldsymbol{r} \tag{12}$$

[6] For instance, this is not true at a point where power density is null.

[7] The CMIR hypothesis, as I stated it, is somewhat imprecise. I defined what a hot spot is only imprecisely, and I did not define what it means for (9) to be approximately true. Vagueness is intentional here.

With $\bar{u}$ as the output of the LTI system, and using the same eigenfunction expansion as Gerstenmaier and Wachutka, Codecasa proves that the impulse response is a *positive* sum of decreasing exponentials [17].

Average temperature $\bar{u}$ is an abstract quantity. Instead, I choose to focus on the temperature θ at a hot spot. Nevertheless, as $\bar{u}$ gives more weight to temperature in high power density regions, it is plausible that the shape of the impulse response for $\bar{u}$ resembles that for θ.

To support the CMIR hypothesis empirically, I modeled a silicon chip and copper heat spreader and heatsink using the ATMI software [51, 4], assuming a uniform disk heat source. I tested 1458 configurations, varying the thickness of the silicon die ($\in \{0.2, 0.5, 1\}$ mm), the thickness of the copper spreader and heatsink base ($\in \{2, 5, 10\}$ mm), the heatsink width ($\in \{5, 7, 10\}$ cm), the heatsink thermal resistance ($\in \{0.1, 0.4, 2\}$ K/W) the thermal interface material (TIM1) resistance ($\in \{0.05, 0.2, 1\}$ Kcm^2/W) and the radius of the heat source ($\in \{0.5, 1, 2, 5, 10, 20\}$ mm). ATMI gives the step response $H(t)$ for each configuration, with the center of the disk source as point $\boldsymbol{r_\theta}$.

Then, I approximate $H(t)$ as in equation (11), with a finite n. The problem is to determine n, the λ_i's and the H_i's.[8] Without loss of generality, we can assume

$$\lambda_1 < \lambda_2 < \ldots < \lambda_n$$

I set the value of λ_1 assuming a lumped thermal capacitance. The time constant $1/\lambda_1$ equals $R_{hs} \times (C_{Cu} + C_{Si})$, where R_{hs} is the heatsink thermal resistance and $C_{Cu} + C_{Si}$ is the total thermal capacitance. I set n to a fixed value, e.g., $n = 20$. Then I chose a large enough λ_n, e.g., $1/\lambda_n = 1\,\mu$s. The other λ_i's are defined from λ_1 and λ_n so that the λ_i's form a geometric sequence. In most cases, n and λ_n do not need to be set precisely.[9]

Finally, I obtained the H_i's by solving a non-negative least squares (NNLS) problem [41], which enforces the constraint $H_i \geq 0$. More precisely, I solved (in the least-square sense) an overdetermined system of $n+1$ linear equations corresponding to equating the true $H(t)$ with the approximate one at the times $t = 1/\lambda_i$ and at $t = \infty$. I used the NNLS software written by Suvrit Sra [39, 53]. On the 1458 tested configurations, with $n = 20$, the difference between the true $H(t)$ and the approximate one never exceeded 2% of $H(\infty)$. In practice, the NNLS algorithm often finds a null or negligible value for several of the H_i's.

2.6 State-space representation

The LTI system described by equation (8) is an infinite-dimensional (aka distributed parameter) system. Indeed, given a time t_0, the evolution of temperature for $t > t_0$ is determined only if we know $p(t)$ for all $t \in [0, t_0]$ or if we know $T(\boldsymbol{r}, t_0)$ for all $\boldsymbol{r} \in \mathcal{V}$. Infinite-dimensional systems can often be approximated with finite-dimensional models. The HotSpot thermal model is an example of finite-dimensional model obtained by discretizing space [32].

Equation (9) with a finite n leads to a finite-dimensional system. By combining (8) and (9), we have

$$\theta = \sum_{i=1}^{n} H_i X_i \tag{13}$$

with

$$X_i := p * (\lambda_i e^{-\lambda_i t}) = \lambda_i e^{-\lambda_i t} \int_0^t p(x) e^{\lambda_i x} dx \tag{14}$$

[8] The problem of extracting multiexponential decay parameters from empirical data is a classical problem, and an ill-posed one [35]. Our problem is closely related but much easier, as parameters n, λ_i and H_i have no physical significance. What matters is that the approximate $H(t)$ is close to the actual $H(t)$.

[9] On a few configurations, λ_n needed some adjustment.

Note that X_i is continuous even if p is not.

For all $i \in \{1, \dots, n\}$, quantity X_i is the solution of the following differential equation:

$$\frac{dX_i}{dt} = \lambda_i(p - X_i) \tag{15}$$

From equation (15), it is clear that the knowledge of all X_i's at a time t_0 and the knowledge of $p(t)$ for $t \geq t_0$ are sufficient to determine $\theta(t)$ for $t \geq t_0$:

$$\theta(t) = \Big[\sum_{i=1}^{n} H_i X_i(t_0) e^{-\lambda_i(t-t_0)}\Big] + \int_{t_0}^{t} p(x)h(t-x)dx \tag{16}$$

In the particular case when p is constant during a time interval $[t, t+\tau)$, solving equation (15) yields

$$X_i(t+\tau) = X_i(t)e^{-\lambda_i\tau} + p(1 - e^{-\lambda_i\tau}) \tag{17}$$

Matrix notation is sometimes used later in this document. Equation (13) can be written

$$\theta = \boldsymbol{H}^T \boldsymbol{X} \tag{18}$$

where $\boldsymbol{X} = (X_1, \dots, X_n)$ is a column vector and $\boldsymbol{H}^T = (H_1, \dots, H_n)$ is a row vector. Equation (15) can be written

$$\frac{d\boldsymbol{X}}{dt} = \boldsymbol{\lambda} p - \mathbf{D}[\boldsymbol{\lambda}]\boldsymbol{X} \tag{19}$$

where $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)$ and $\mathbf{D}[\boldsymbol{\lambda}] = \mathbf{D}[\lambda_1, \dots, \lambda_n]$ is the $n \times n$ diagonal matrix with the λ_i's on the diagonal. Equations (18) and (19) define a state-space representation [16]. Unlike finite-dimensional models obtained by discretizing space, the X_i's have no direct physical meaning.[10]

In matrix notation, equation (17) can be written

$$\boldsymbol{X}(t+\tau) = e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}\boldsymbol{X}(t) + p(\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau})\mathbf{1} \tag{20}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{1} = (1, \dots, 1)$ is a column vector. Note that a matrix exponential is being used [50].

Finally, note that equation (17) acts like an exponentially weighted moving average (EWMA). In the degenerate case $n = 1$, θ is proportional to X_1. The turbo control implemented in the Intel Sandy Bridge is based on an order-one thermal model, i.e., one thermal resistance and one thermal capacitance. This explains the EWMA used to calculate the energy credit in the Sandy Bridge [64, 65]. The turbo control proposed in this document can be viewed as a generalization of the Sandy Bridge turbo.

3 Maximizing processor performance

This section considers the problem of maximizing processor performance. Cohen et al. showed that, contrary to intuition, maximizing performance is not always equivalent to maximizing temperature [22]. Rao et al., independently, reached a similar conclusion [59].

My motivations for revisiting this problem are:

- Even though maximizing temperature is not, strictly speaking, the optimal control strategy under dynamic voltage/frequency scaling (DVFS), it would be interesting to know whether it is close to optimal.

[10]See for instance the discussion in [6] about modeling heat conduction with a thermal Foster network.

- Both Cohen and Rao assumed an order-one thermal model consisting of a single resistance and a single capacitance.
- Neither Cohen nor Rao considers the dependence of leakage on temperature.

I show that, for the SISO temperature model of Section 2, if power is an affine function of the clock frequency and of the temperature output, then maximizing performance is equivalent to maximizing temperature.

3.1 Affine power model

The turbo control sets $p(t)$ only indirectly, via the voltage and the clock frequency. Actually, voltage and frequency are not independent. To reduce power, the turbo control sets voltage to the lowest safe value for the target frequency. In practice, voltage is approximately an affine function of frequency:

$$v = v_0 + (1 - v_0)f \tag{21}$$

For example, using the data provided in [58, 30, 28] for three different Intel processors and applying a least-square fit, we get $v_0 \simeq 0.6$. I assume perfect DVFS, that is, I neglect the transition delay for changing voltage and assume that v is a function of f, making f the sole input.

Power is a somewhat complicated function of frequency and temperature. To simplify the maths, instead of using equation (7), I assume that power can be approximated as an affine function of f and θ:

$$p = [\alpha + \beta f + \gamma\theta]P_w \tag{22}$$

with $P_w = P_s + P_d$ constant, $\beta > 0$ and $\gamma > 0$.

I believe that (22) is a reasonable approximation for realistic frequency and temperature ranges. For example, assuming $v_0 = 0.6$ in equation (21) and using the leakage power model described in [44], with $T_{amb} = 40$ °C, $T_{max} = 100$ °C, maximum voltage 1 V and $P_s/P_w = 0.3$, I find that taking $\alpha = -0.31$, $\beta = 1.18$ and $\gamma = 0.0017$ gives a maximum error of about 8% for $f \in [0.5, 1]$ and $\theta \in [0, 60]$.

3.2 How temperature varies with clock frequency

The processor alternates activity and inactivity phases. Let us focus on one particular activity phase starting at time t_0, with initial state $\boldsymbol{X}(t_0)$ resulting from past processor activity. Defining $\tau = t - t_0$, equation (16) becomes

$$\theta = \theta_0(\tau) + \int_0^\tau p(t_0 + x)h(\tau - x)dx$$

with

$$\theta_0(\tau) = \sum_{i=1}^{n} H_i X_i(t_0)e^{-\lambda_i \tau}$$

Replacing $p(t_0 + x)$ with $[\alpha + \beta f + \gamma\theta]P_w$ and considering θ and f as functions of τ yields the following integral equation:

$$\theta = \theta_0 + P_w[\alpha + \beta f + \gamma\theta] * h \tag{23}$$

Let us consider a frequency function $f(\tau)$ yielding temperature $\theta(\tau)$ according to equation (23), and let us consider a frequency $f(\tau) + \delta f(\tau)$ yielding temperature $\theta(\tau) + \delta\theta(\tau)$:

$$\theta + \delta\theta = \theta_0 + P_w[\alpha + \beta(f + \delta f) + \gamma(\theta + \delta\theta)] * h \tag{24}$$

where $\delta\theta$ is the temperature variation resulting from frequency variation δf. Subtracting (23) from (24) gives

$$\delta\theta = P_w[\beta\delta f + \gamma\delta\theta] * h \tag{25}$$

One can easily verify that equation (25) defines an LTI system, with $\beta\delta f$ as input and $\delta\theta$ as output:

$$\delta\theta = \beta\delta f * k \tag{26}$$

where $k(\tau)$ is the impulse response of that system.

3.3 Characterizing the impulse response

The transfer function $\hat{k}(s)$ is the Laplace transform of the impulse response $k(\tau)$:

$$\hat{k}(s) := \int_0^\infty k(\tau)e^{-s\tau}d\tau \tag{27}$$

where s is complex. Using the fact that convolution in the time domain is a product in the Laplace domain [16, 54], equation (25) becomes, in the Laplace domain:

$$\delta\hat{\theta} = P_w[\beta\delta\hat{f} + \gamma\delta\hat{\theta}] \times \hat{h}$$

which yields

$$\hat{k} = \frac{\delta\hat{\theta}}{\beta\delta\hat{f}} = \frac{P_w\hat{h}}{1-\gamma P_w\hat{h}} \tag{28}$$

For the system to be stable, the poles of the transfer function must have negative real part. Note that, for s with non-negative real part, $|\hat{h}(s)| \leq \int_0^\infty |h(\tau)e^{-s\tau}|d\tau \leq \hat{h}(0) = H(\infty)$. Consequently, the system is stable if

$$\gamma P_w H(\infty) < 1 \tag{29}$$

If condition (29) were not true, thermal runaway could happen [80]. Therefore I assume that condition (29) holds.

It is easy to see why $h \geq 0$ implies $k \geq 0$. Indeed, equation (28), in the time domain, is a Volterra integral equation of the second kind:

$$k = P_w h + \gamma P_w h * k$$

The solution k can be obtained by successive approximation, taking the limit [45, 84]: $k_0 = P_w h$ and $k_{i+1} = P_w h + \gamma P_w h * k_i$. The positivity of h implies that of k_0, k_1, k_2, and so on.

Less obvious is the fact that k inherits complete monotonicity from h, provided that condition (29) holds. The fact that the right-hand side (RHS) of equation (28) is the Laplace transform of a completely monotone function was proved by Keilson for a problem in queueing theory [38]. Let us rewrite (28) by normalizing $\hat{h}$ and $\hat{k}$ so that they can be viewed as Laplace transforms of some probability density functions:

$$\frac{\hat{k}}{\hat{k}(0)} = \frac{[1-\gamma P_w H(\infty)]\frac{\hat{h}}{\hat{h}(0)}}{1-\gamma P_w H(\infty)\frac{\hat{h}}{\hat{h}(0)}}$$

Then, complete monotonicity of h, condition (29) and theorem 3.1 in [38] establish the complete monotonicity of k. Alternatively, the complete monotonicity of k can be proved from theorem 5.2.6 in [29].

3.4 Maximizing performance

Maximizing performance means maximizing the work done in a given time. If we define one clock cycle as the unit of work, the work done up to time τ is

$$F(\tau) := \int_0^\tau f(x)dx$$

Quantity $\delta F = \int_0^\tau \delta f(x)dx$ is the variation of work resulting from the variation of frequency δf. Let us rewrite equation (26) using integration by parts:

$$\frac{\delta\theta(\tau)}{\beta} = \int_0^\tau \delta f(x)k(\tau - x)dx = k(0)\delta F(\tau) + \int_0^\tau \delta F(x)k'(\tau - x)dx$$

where k' is the derivative of k. We obtain the following Volterra integral equation of the second kind:

$$\delta F = \frac{\delta\theta}{\beta k(0)} - \frac{k'}{k(0)} * \delta F \tag{30}$$

Equation (30) can be solved for δF by successive approximation [45, 84]: $\delta F_0 = \frac{\delta\theta}{\beta k(0)}$ and $\delta F_{i+1} = \frac{\delta\theta}{\beta k(0)} - \frac{k'}{k(0)} * \delta F_i$. The complete monotonicity of k implies $k' \leq 0$. Therefore, $\delta\theta(x) \geq 0$ for $x \in [0, \tau]$ implies $\delta F(x) \geq 0$ for $x \in [0, \tau]$. In other words, maximizing temperature maximizes performance.

This conclusion seemingly contradicts the findings of Cohen and Rao [22, 59]. The discrepancy lies in the assumption that power is an affine function of frequency and temperature. Strictly speaking, maximizing temperature is not the optimal strategy. In practice though, it is close to optimal as long as clock frequency lies in an interval where equation (22) is an accurate model.

4 A turbo control for a SISO system

As shown in Section 3, a simple strategy for maximizing performance under an affine power model is to maximize temperature.[11] There are three different regimes:

- **sprint**: as long as temperature is less than T_{max}, run at the maximum clock frequency and voltage.
- **thermal saturation**: when temperature reaches T_{max}, adjust frequency and voltage so as to stay at T_{max}.
- **relaxation**: the processor is inactive.

As mentioned in Section 1, the requirement that performance be deterministic means that the turbo control cannot rely on thermal sensors but must use a calculated temperature. Power too may have to be calculated during thermal saturation so that the calculated temperature stays as close to T_{max} as possible. This section describes those calculations.

[11]The proposed turbo control does not rely on any particular power model. Nevertheless, the proposed turbo control is not necessarily close to optimal if the power function is not approximately affine.

4.1 Sprint and relaxation

For a SISO system, it is not necessary to calculate temperature during relaxation. It is sufficient to update the X_i's at the end of the relaxation phase by setting τ equal to whatever time has elapsed since the processor became inactive, putting $p = 0$ into equation (17).

During a sprint, relative temperature θ can be calculated from equations (13) and (17) using a fixed timestep τ and assuming that power p is constant during each time interval $[t, t+\tau)$. The X_i's are updated with equation (17) on every timestep. The sprint ends whenever the calculated temperature θ reaches

$$\theta_{max} := T_{max} - T_{amb},$$

at which point the thermal saturation regime starts.

4.2 Thermal saturation

In the sprint and relaxation regimes, power p is either null or is obtained from equation (22). In the thermal saturation regime, we must calculate p so that θ stays close to θ_{max}. It is assumed that, at the time t when thermal saturation starts, $\theta(t) \simeq \theta_{max}$. Then, it is sufficient for the turbo control to keep temperature constant. Two alternative methods are proposed.

4.2.1 Discrete method

A possible method for keeping temperature constant is to use a fixed timestep τ and to apply a constant power p during $[t, t+\tau)$ such that $\theta(t+\tau) = \theta(t)$. From (13) and (17), we have

$$\theta(t+\tau) = \sum_{i=1}^{n} H_i X_i(t+\tau) = H(\tau)p + \sum_{i=1}^{n} H_i X_i(t) e^{-\lambda_i \tau} \tag{31}$$

We can have $\theta(t+\tau) = \theta(t)$ by setting p as follows:

$$p = \frac{\theta(t) - \sum_{i=1}^{n} H_i X_i(t) e^{-\lambda_i \tau}}{H(\tau)} = \frac{\sum_{i=1}^{n} H_i (1 - e^{-\lambda_i \tau}) X_i(t)}{\sum_{i=1}^{n} H_i (1 - e^{-\lambda_i \tau})} \tag{32}$$

Note that p is a weighted arithmetic mean of the X_i's. This method does not require that $H(t)$ be a Bernstein function. However, it requires $H(\tau) \neq 0$. The turbo control computes the power curve by iterating consecutive timesteps, updating the X_i's with equation (17) and the calculated p value.

4.2.2 Continuous method

In this section, it is assumed that the h_i's are non-null. That is, n is the number of non-null h_i's obtained with the NNLS method of Section 2.5.

A possible method for keeping temperature constant is to search for the power p such that $\frac{d\theta}{dt} = 0$. From (13) and (15), we have

$$\frac{d\theta}{dt} = \sum_{i=1}^{n} H_i \frac{dX_i}{dt} = h(0)p - \sum_{i=1}^{n} h_i X_i$$

Therefore, the power p such that $\frac{d\theta}{dt} = 0$ is

$$p = \frac{1}{h(0)} \sum_{i=1}^{n} h_i X_i = \frac{1}{h(0)} \boldsymbol{h}^T \boldsymbol{X} \tag{33}$$

Note that (33) is the limit of (32) when $\tau \to 0$. Note also that the continuity of the X_i's implies that of p during thermal saturation.

Introducing (33) into (19), we obtain a system of differential equations in matrix form:

$$\frac{d\boldsymbol{X}}{dt} = -\mathbf{M}\boldsymbol{X} \tag{34}$$

where matrix $\mathbf{M}$ is

$$\mathbf{M} := \mathbf{D}[\boldsymbol{\lambda}] - \frac{1}{h(0)}\boldsymbol{\lambda}\boldsymbol{h}^T \tag{35}$$

The solution of equation (34) can be expressed with a matrix exponential:

$$\boldsymbol{X}(t+\tau) = e^{-\mathbf{M}\tau}\boldsymbol{X}(t) \tag{36}$$

The matrix exponential can be calculated by diagonalizing matrix $\mathbf{M}$. The eigenvalues μ of $\mathbf{M}$ are the solutions of the characteristic equation

$$\det(\mathbf{M} - \mu\mathbf{I}) = 0 \tag{37}$$

Putting (35) into (37), eigenvalues μ are the solutions of

$$\det\Big(\mathbf{D}[\boldsymbol{\lambda}] - \frac{1}{h(0)}\boldsymbol{\lambda}\boldsymbol{h}^T - \mu\mathbf{I}\Big) = 0 \tag{38}$$

To simplify the equation, we can use the following identity for a rank-one perturbation of an invertible matrix $\mathbf{A}$ (equation 6.2.3 in Meyer's book [50]):

$$\det(\mathbf{A} + \boldsymbol{c}\boldsymbol{d}^T) = (1 + \boldsymbol{d}^T\mathbf{A}^{-1}\boldsymbol{c})\det(\mathbf{A})$$

where $\boldsymbol{c}$ is a column vector and $\boldsymbol{d}^T$ is a row vector. Assuming $\mu \neq \lambda_i$ and applying the identity above to the diagonal matrix $\mathbf{A} = \mathbf{D}[\boldsymbol{\lambda}] - \mu\mathbf{I}$, equation (38) becomes

$$1 - \frac{1}{h(0)}\boldsymbol{h}^T\mathbf{D}\Big[\frac{1}{\lambda_1 - \mu}, \dots, \frac{1}{\lambda_n - \mu}\Big]\boldsymbol{\lambda} = 0$$

or equivalently

$$\sum_{i=1}^{n} \frac{\lambda_i h_i}{\lambda_i - \mu} = h(0) = \sum_{i=1}^{n} h_i$$

Finally, the characteristic equation for matrix $\mathbf{M}$ is simply

$$\mu \sum_{i=1}^{n} \frac{h_i}{\lambda_i - \mu} = 0 \tag{39}$$

Complete monotonicity of $h(t)$, that is, $h_i > 0$, implies that equation (39) has n distinct solutions $\mu_1, \dots, \mu_n$:

$$\begin{aligned} \mu_1 &= 0 \\ \lambda_{i-1} < \mu_i &< \lambda_i \quad \text{for } i \in [2, n] \end{aligned}$$

Therefore, matrix M is diagonalizable:

$$\mathbf{M} = \mathbf{Q}\mathbf{D}[\boldsymbol{\mu}]\mathbf{Q}^{-1} \tag{40}$$

where the columns of matrix $\mathbf{Q}$ are right eigenvectors of $\mathbf{M}$ and the rows of $\mathbf{Q}^{-1}$ are left eigenvectors of $\mathbf{M}$. The eigenvectors can be obtained directly from the eigenvalues. Let $\boldsymbol{x_\mu} = (x_1, \ldots, x_n)$ be a right eigenvector associated with eigenvalue μ, that is,

$$\mathbf{M}\boldsymbol{x_\mu} = \mu\boldsymbol{x_\mu}$$

Elementwise, and introducing (35) into it, the above equation is equivalent to

$$\lambda_i x_i - \frac{1}{h(0)}\lambda_i \boldsymbol{h}^T \boldsymbol{x_\mu} = \mu x_i$$

that is,

$$\frac{(\lambda_i - \mu)x_i}{\lambda_i} = \frac{1}{h(0)}\boldsymbol{h}^T \boldsymbol{x_\mu} \tag{41}$$

The RHS of equation (41) is the same for all $i \in \{1, \ldots, n\}$ and can be set to an arbitrary non-null value (the eigenspaces have dimension one). For instance, setting the RHS of (41) equal to 1, we obtain $x_i = \frac{\lambda_i}{\lambda_i - \mu}$, that is, the element in the i^{th} row and j^{th} column of matrix $\mathbf{Q}$ is

$$[\mathbf{Q}]_{ij} = \frac{\lambda_i}{\lambda_i - \mu_j} \tag{42}$$

Left eigenvectors can be obtained in a similar way: if $\boldsymbol{y_\mu}^T = (y_1, \ldots, y_n)$ is a left eigenvector associated with eigenvalue μ, then for all $j \in \{1, \ldots, n\}$,

$$\frac{(\lambda_j - \mu)y_j}{h_j} = \frac{1}{h(0)}\boldsymbol{y_\mu}^T \boldsymbol{\lambda}$$

The RHS is the same for all $j \in \{1, \ldots, n\}$ and is chosen such that $\boldsymbol{y_\mu}^T \boldsymbol{x_\mu} = 1$. Defining

$$\frac{1}{\sigma_i} := \sum_{j=1}^{n} \frac{\lambda_j h_j}{(\lambda_j - \mu_i)^2}$$

we have

$$[\mathbf{Q}^{-1}]_{ij} = \frac{\sigma_i h_j}{\lambda_j - \mu_i} \tag{43}$$

Finally, the matrix exponential in (36) can be calculated from (40) as

$$e^{-\mathbf{M}\tau} = \mathbf{Q}e^{-\mathbf{D}[\boldsymbol{\mu}]\tau}\mathbf{Q}^{-1} = \mathbf{Q}\mathbf{D}[e^{-\mu_1\tau}, \ldots, e^{-\mu_n\tau}]\mathbf{Q}^{-1} \tag{44}$$

That is, the element in the i^{th} row and j^{th} column of matrix $e^{-\mathbf{M}\tau}$ is

$$[e^{-\mathbf{M}\tau}]_{ij} = \lambda_i h_j \sum_{l=1}^{n} \frac{\sigma_l e^{-\mu_l \tau}}{(\lambda_i - \mu_l)(\lambda_j - \mu_l)} \tag{45}$$

As a sanity check when calculating matrix $e^{-\mathbf{M}\tau}$, it should be noted that $e^{-\mathbf{M}\tau}$ is a *stochastic* matrix [50]:

$$\forall i, \quad \sum_{j=1}^{n} [e^{-\mathbf{M}\tau}]_{ij} = 1$$

$$\forall \tau > 0,\ \forall i,j, \quad [e^{-\mathbf{M}\tau}]_{ij} > 0$$

The positivity of matrix $e^{-\mathbf{M}\tau}$ for $\tau > 0$ comes from $-\mathbf{M}\tau$ being an irreducible Metzler matrix.[12]

The power keeping temperature equal to $\theta(t)$ is calculated from (36) and (33):

$$p(t+\tau) = \frac{\theta(t)}{H(\infty)} + \frac{1}{h(0)} \sum_{l=2}^{n} \Big[\sum_{i=1}^{n} \frac{\lambda_i h_i}{\lambda_i - \mu_l}\Big] \Big[\sum_{j=1}^{n} \frac{h_j X_j(t)}{\lambda_j - \mu_l}\Big] \sigma_l e^{-\mu_l \tau} \tag{46}$$

where we have used the fact that $\mu_1 = 0$ and $\sigma_1 = 1/H(\infty)$.

Unlike the discrete method, which calculates the power curve one timestep at a time, the continuous methods calculates the whole power curve at once. The continuous method is called so because, after an initial discontinuity when thermal saturation starts, power (46) varies continuously afterwards. However, whether we use the discrete method or the continuous method, the clock frequency can only take discrete values. In practice, DVFS control tries to follow (46) as closely as possible, within its own constraints. State $\boldsymbol{X}$ is updated with equation (36) at the time $t + \tau$ when power p deviates significantly from (46), e.g., when processor activity ceases.

5 Worst-case step response

The two main assumptions of the SISO model, i.e., a power density of the form (4) and the assumption that the turbo control need only consider temperature at a single point, lead to a single step response. However, different workloads may generate different power density maps and different step responses. Moreover, even for a given power density map, distinct regions of the chip may experience differently shaped step responses. In particular, the chip region where steady-state temperature is the highest is not necessarily the region where transient temperature increases the fastest.

A possible way to take into account multiple step responses is to define a single worst-case step response. A turbo control using such step response will generally overestimate temperature and reduce processor performance. However, this approach preserves performance determinism.

First, one should collect step responses by running some “hot” workloads on a prototype platform or on a calibrated simulator.[13] Step responses that are consistently less than some other step responses can be removed from the collection. That is, all the step responses in the collection cross each other at one or several points.

Then, a worst-case step response is calculated by taking the *maximum* of all the step responses in the collection. The discussion below considers two different methods for defining the maximum of two step responses.[14]

The most natural way to define the maximum $H^{**}(t)$ of two functions $H(t)$ and $H^*(t)$ is, for all t,

$$H^{**}(t) := \max(H, H^*)(t) := \max(H(t), H^*(t)) \tag{47}$$

Then, the NNLS method described in Section 2.5 can be applied to H^{**}. Note however that, even if H and H^* are Bernstein functions, H^{**} is not necessarily a smooth function, let alone a Bernstein one. So the Bernstein function obtained with the NNLS method might depart significantly from the actual H^{**} and is not guaranteed to be consistently greater than or equal to H and H^*.

[12]For $\tau > 0$, we have $(-\mathbf{M}+\lambda_n\mathbf{I})\tau > 0$, hence $e^{(-\mathbf{M}+\lambda_n\mathbf{I})\tau} > 0$. As $\mathbf{I}$ commutes with any matrix, $e^{(-\mathbf{M}+\lambda_n\mathbf{I})\tau} = e^{-\mathbf{M}\tau}e^{\lambda_n\mathbf{I}\tau} = e^{-\mathbf{M}\tau}\mathbf{I}e^{\lambda_n\tau} = e^{-\mathbf{M}\tau}e^{\lambda_n\tau} > 0$. Therefore, $e^{-\mathbf{M}\tau} > 0$. See also Theorem 2.7 in [71].

[13]Thermal interface material (TIM) characteristics may degrade with usage [68]. A worst-case TIM should probably be assumed when collecting step responses.

[14]Both methods are commutative and associative. They generalize unambiguously to multiple step responses.

parameter	unit	value
silicon thermal conductivity (70 °C)	W/(mK)	125
silicon thermal diffusivity (70 °C)	m^2/s	7×10^{-5}
silicon thickness	mm	0.5
thermal interface resistance (TIM1)	Kcm^2/W	0.2
copper thermal conductivity	W/(mK)	400
copper thermal diffusivity	m^2/s	1.16×10^{-4}
copper thickness	mm	3
heatsink width	cm	5
heatsink thermal resistance	K/W	0.6

Table 2: ATMI parameters (see [51]). Thermal diffusivity equals κ/c_v. TIM1 resistance per cm^2 equals $10z_i/\kappa_i$ with z_i the thickness in millimeter of the interface material and κ_i its thermal conductivity in W/(mK).

Alternatively, the maximum of two step responses can be defined as follows. Given a function $H(t) = \sum_{i=1}^{n} H_i(1 - e^{-\lambda_i t})$ with $0 < \lambda_1 < \lambda_2 < \ldots < \lambda_n$, let us define its *dual* function $S(x)$ for $x \geq 0$:

$$S(x) := \sum_{\lambda_i \geq x} H_i \tag{48}$$

with $S(x) = 0$ for $x > \lambda_n$. Note that $S(x)$ is piecewise constant with discontinuities at the λ_i's:

$$S(\lambda_i) - S(\lambda_i^+) = H_i$$

One can easily verify that

$$H(t) = t \int_0^\infty S(x) e^{-xt} dx \tag{49}$$

The maximum S^{**} of two dual functions S and S^* is

$$S^{**}(x) := \max(S, S^*)(x) := \max(S(x), S^*(x)) \tag{50}$$

Note that S^{**} is piecewise constant, its discontinuities $\{\lambda_1^{**}, \ldots, \lambda_{n^{**}}^{**}\}$ being a subset of $\{\lambda_1, \ldots, \lambda_n, \lambda_1^*, \ldots, \lambda_{n^*}^*\}$. Hence the function H^{**} whose dual is S^{**} is a finite sum of decreasing exponentials. It is this function H^{**} that can be defined as the maximum of H and H^*.

Some properties of definition (50):

- $H^{**}(t) \geq \max(H(t), H^*(t))$ for all t (from (50) and (49)).
- The maximum of two Bernstein functions is a Bernstein function. Indeed, a function is Bernstein iff its dual is monotonically decreasing. If S and S^* are both monotonically decreasing, so is S^{**}.
- $H^{**}(\infty) = \max(H(\infty), H^*(\infty))$ (from $S(0) = H(\infty)$). In other words, taking the maximum of multiple step responses does not increase the maximum junction-to-ambient thermal resistance.
- If the same set of λ_i's is common to H and H^*, it is also the set of λ_i's for H^{**}.

To illustrate definition (50), I collected some step responses with the ATMI thermal model [51]. I used the parameters listed in Table 2 and the example power density map shown in Figure 1. Table 3 provides parameters λ_i and H_i's for the step responses H and H^* at the center of each

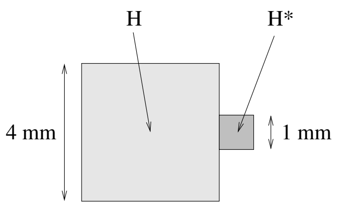


Figure 1: Example power density map. Power density is uniform in each square, but the power density of the small square is twice that of the large square. H and H^* are the step responses at the center of the large and small squares, respectively.

	i	1	2	3	4	5	6	7	8
H	λ_i	0.0593	1.22	25.2	114	10700			
	H_i	0.59	0.10	0.46	0.64	0.04			
H^*	λ_i^*	0.0593	1.22	5.55	25.2	114	519	2360	10700
	H_i^*	0.60	0.07	0.07	0.25	0.48	0.02	0.14	0.05
H^{**}	λ_i^{**}	0.0593	1.22	25.2	114	519	2360	10700	
	H_i^{**}	0.59	0.10	0.46	0.48	0.02	0.14	0.05	

Table 3: Parameters for step responses H and H^* were obtained as described in Section 2.5. H^{**} is the maximum of H and H^* according to definition (50).

square. I obtained the parameter values with NNLS, as described in Section 2.5, but starting from a geometric sequence of 12 λ_i values. I ignored H_i's whose value is less than 1% of $H(\infty)$, which yields $n = 5$ for H and $n^* = 8$ for H^*. Table 3 also shows the parameters for H^{**}, the maximum of H and H^* according to definition (50).

Figure 2 shows the step responses H, H^* and H^{**} corresponding to the parameters given in Table 3. Notice that H^* increases faster than H for small times, as it corresponds to a region of higher power density. However, after about 10 ms, H exceeds H^* and converges to a higher steady-state value. H^{**} is always above H^* and H, coinciding with H^* at small times and with H at large times.

It should be noted that, whether definition (47) or (50) is used, the maximum of multiple step responses corresponding to different power density maps and different sensor locations, is an abstract step response that does not correspond to any particular power density map or any particular sensor location.

6 Multi-input multi-output (MIMO) system

The assumptions of the SISO model are not valid for multicore chips where cores may be active or inactive independently of each other. This section proposes a more accurate multi-input multi-output (MIMO) temperature model based on the following power density:

$$g(\boldsymbol{r},t) = \sum_{j=1}^{N} m_j(\boldsymbol{r})p_j(t) \tag{51}$$

where N is the number of cores and p_j is the power dissipation contributed by core j.

The temperature generated by power density (51) can be calculated from the superposition principle (3), by considering separate impulse responses corresponding to each power density

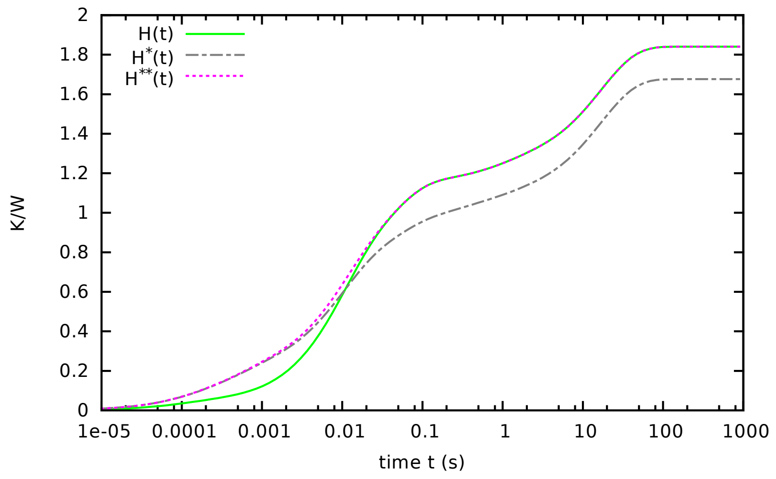


Figure 2: H^{**} is the maximum of the step responses H and H^* according to definition (50).

maps m_j. Temperature θ_i in core i is

$$\theta_i(t) = \sum_{j=1}^{N} \theta_{ij}(t) \tag{52}$$

where θ_{ij} is the output of the SISO system with p_j as input and step response

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

$\boldsymbol{r}_{\boldsymbol{\theta_i}}$ being the point in core i where temperature is being monitored.

Note that model (51) assumes fixed power density maps $m_j(\boldsymbol{r})$. However, as already mentioned in Section 5, actual power density maps depend on workload characteristics. Step responses $H_{ij}(t)$ for $i \neq j$ are only weakly dependent on $m_j(\boldsymbol{r})$ and $\boldsymbol{r}_{\boldsymbol{\theta_i}}$, especially if cores i and j are distant from each other. For self-heating step responses $H_{ii}(t)$, the method of worst-case step response described in Section 5 can be used.

Some remarks:

- For $i \neq j$, $H_{ij}(t)$ is not a Bernstein function. This is because it takes some time for heat to diffuse from core j to core i. Hence $h_{ij}(0) = 0$, which means that the impulse response cannot be both positive and monotonically decreasing, which is a necessary condition for complete monotonicity. Nevertheless, $h_{ij}(t)$ can still be expressed as a linear combination of decreasing exponentials, although some parameters h_i's in (9) are negative. It should be noted that $H_{ij}(t)$ is the difference between two Bernstein step responses (as $m_j = (m_i + m_j) - m_i$), and the NNLS method described in Section 2.5 permits obtaining the parameters for these two Bernstein step responses, hence for $H_{ij}(t)$.

- The CMIR hypothesis was needed only in sections 3.4 and 4.2.2. The state-space representation of Section 2.6 applies regardless of the sign of the H_i's in equation (11). Hence we can calculate θ_{ij} by maintaining a state $\boldsymbol{X_{ij}}$.

- If all the step responses use a common $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)$, it is sufficient to maintain N state vectors $\boldsymbol{X_j} = \boldsymbol{X_{ij}}$, that is, one state vector per core. From (18),

$$\theta_{ij} = \boldsymbol{H_{ij}}^T \boldsymbol{X_{ij}} = \boldsymbol{H_{ij}}^T \boldsymbol{X_j}$$

When a constant power p_j is dissipated by core j during a time interval $[t, t+\tau)$, the state vector $\boldsymbol{X_j}$ can be updated with equation (20), that is,

$$\boldsymbol{X_j}(t+\tau) = e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}\boldsymbol{X_j}(t) + p_j(\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau})\mathbf{1} \tag{53}$$

We have

$$\begin{aligned}
\theta_i(t+\tau) - \theta_i(t) &= \sum_{j=1}^{N}[\theta_{ij}(t+\tau) - \theta_{ij}(t)] \\
&= \sum_{j=1}^{N}\boldsymbol{H_{ij}}^T[\boldsymbol{X_j}(t+\tau) - \boldsymbol{X_j}(t)] \\
&= -\sum_{j=1}^{N}\boldsymbol{H_{ij}}^T[\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}]\boldsymbol{X_j}(t) + \sum_{j=1}^{N}p_j\boldsymbol{H_{ij}}^T[\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}]\mathbf{1} \\
&= -\Delta_i(t,\tau) + \sum_{j=1}^{N}H_{ij}(\tau)p_j
\end{aligned} \tag{54}$$

with $\Delta_i(t,\tau)$ defined as

$$\Delta_i(t,\tau) := \sum_{j=1}^{N}\boldsymbol{H_{ij}}^T[\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}]\boldsymbol{X_j}(t) \tag{55}$$

Equation (54) can be used to update the θ_i's at every timestep. It can also be used to get an estimate of $\theta_i(t+\tau)$ from estimated p_j's. In particular, if τ is less than the time for heat to diffuse from a core to another core, $H_{ij}(\tau)$ is small for $i \neq j$, and we can use the following estimate:

$$\theta_i(t+\tau) - \theta_i(t) \simeq -\Delta_i(t,\tau) + H_{ii}(\tau)p_i \tag{56}$$

Equation (56) says that, if τ is small enough, $\theta_i(t+\tau) - \theta_i(t)$ mostly depends on p_i. However, recall that equation (56) is just an approximation.[15]

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

$$\boldsymbol{\theta} = \mathbf{R}\boldsymbol{p} \tag{57}$$

where $\mathbf{R}$ is the $N \times N$ matrix such that

$$[\mathbf{R}]_{ij} = R_{ij} := H_{ij}(\infty) = \boldsymbol{H_{ij}}^T\mathbf{1}$$

Obviously, matrix $\mathbf{R}$ is positive ($R_{ij} > 0$). Furthermore, as explained in Appendix B, $\mathbf{R}$ is likely to be almost symmetric ($R_{ij} \simeq R_{ji}$), positive definite and well-conditioned. This is what I assume in the rest of the document.

[15]For $i \neq j$, the value $H_{ij}(\tau) = \boldsymbol{H_{ij}}^T[\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}]\mathbf{1}$ is small but non-null due to (11) being an approximation or reality.

7 A turbo control for a MIMO system

This section proposes a turbo control based on the MIMO system of Section 6. Per-core DVFS is assumed. That is, each core has its own clock frequency and voltage [11].

When independent tasks run simultaneously on the same chip, a certain fairness in the use of shared resources must be imposed [79]. Power is a shared resource, and the turbo control must take fairness into account.

I propose that fairness be controlled by a power vector $\boldsymbol{p^*} = (p_1^*, \ldots, p_N^*)$ set by the firmware. Vector $\boldsymbol{p^*}$ is redefined every time 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 $\boldsymbol{\theta^*}$ be the vector of steady-state temperatures corresponding to $\boldsymbol{p^*}$, that is (from (57)),

$$\boldsymbol{\theta^*} = \mathbf{R}\boldsymbol{p^*}$$

The firmware sets the p_i^*'s of active cores such that

$$\max\{\theta_1^*, \ldots, \theta_N^*\} \leq \theta_{max}$$

I define *steady-state power assignment* (SSPA) as follows:

> if $\boldsymbol{p^*}$ remains constant during a time long enough to reach thermal steady state and if power values $\boldsymbol{p^*}$ are attainable, the steady-state $\boldsymbol{p}$ must be equal or close to $\boldsymbol{p^*}$.

Determining how the p_i^*'s of active cores should be set is out of the scope of this document.[16] The question considered here is, given some $\boldsymbol{p^*}$, how can a turbo control take advantage of temperature transients while enforcing SSPA.

To enforce SSPA, the turbo control should take $\boldsymbol{p^*}$ into account when determining the voltage and clock frequency of active cores. However, setting $\boldsymbol{p}$ equal to $\boldsymbol{p^*}$ right away would not yield an interesting turbo control. First, this would not guarantee $\theta_i(t) \leq \theta_{max}$. Second, this would not exploit temperature transients. Instead, I propose the following turbo control.

A fixed timestep τ is assumed. On every timestep, the turbo control calculates a *target* p_j for each core j. Quantity $p_j \times \tau$ is the amount of energy that core j is allowed to dissipate during the next timestep. The frequency and voltage of each core are adjusted (possibly multiple times per timestep) so that, at the end of the timestep, the actual p_j is as close to the target p_j as possible.[17] At the end of the timestep, the $\boldsymbol{X_j}$'s are updated with the *actual* p_j's (equation (53)).

The target p_j's are calculated by solving the linear system

$$\mathbf{B}\boldsymbol{p} = \boldsymbol{Y} \tag{58}$$

where $\mathbf{B}$ is an $N \times N$ matrix and $\boldsymbol{Y} = (Y_1, \ldots, Y_N)$ is a column vector. Matrix $\mathbf{B}$ and vector $\boldsymbol{Y}$ are set as shown in Table 4 (equation (54) is being used here).

The estimated power of core i if core i were to sprint (i.e., run at the maximum frequency) during $[t, t+\tau)$ is denoted π_i. Note that π_i is a prediction, based for instance on the workload's behavior during the previous timestep.

[16]For instance, it makes sense to assign the same p_i^*'s to identical cores. However, the answer is less obvious when cores are different. For instance, how much power should be allotted to a GPU core depending on the number of active CPU cores?

[17]The actual p_j may differ from the target p_j for several reasons: because of constraints besides temperature (voltage/frequency range, power, di/dt, etc.), or because of discrete frequency levels, or if a task completes or starts in the middle of a timestep, or if the workload's behavior changes, etc.

condition on core i	$[\mathbf{B}]_{ij}$	Y_i	regime	result
inactive	δ_{ij}	p_i^*	relaxation	$p_i(t) = p_i^*$
$\theta_i(t) < \theta_i^* - \eta$ $\underline{\text{and}}$ $\tilde{\theta}_i(t+\tau) < \theta_i^* - \eta$	δ_{ij}	π_i	sprint	$p_i(t) = \pi_i$
$\theta_i(t) < \theta_i^* - \eta$ $\underline{\text{and}}$ $\tilde{\theta}_i(t+\tau) \geq \theta_i^* - \eta$	$H_{ij}(\tau)$	$\theta_i^* - \theta_i(t) + \Delta_i(t,\tau)$	saturation	$\theta_i(t+\tau) = \theta_i^*$
$\theta_i^* - \eta \leq \theta_i(t) \leq \theta_i^*$	$H_{ij}(\tau)$	$\theta_i^* - \theta_i(t) + \Delta_i(t,\tau)$	saturation	$\theta_i(t+\tau) = \theta_i^*$
$\theta_i(t) > \theta_i^*$	$H_{ij}(\tau)$	$\epsilon[\theta_i^* - \theta_i(t)] + \Delta_i(t,\tau)$	descent	$\theta_i(t+\tau) < \theta_i(t)$

Table 4: Turbo control for a MIMO system. The p_j's are obtained by solving the linear system (58). $\tilde{\theta}_i(t+\tau)$ is defined in (59). π_i is the power of core i when its clock frequency is maximum. δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ if $i = j$, $\delta_{ij} = 0$ if $i \neq j$). Parameters η and ϵ are positive.

The approximate temperature $\tilde{\theta}_i(t+\tau)$ if core i were to sprint during $[t, t+\tau)$ is (see equation (56))

$$\tilde{\theta}_i(t+\tau) := \theta_i(t) - \Delta_i(t,\tau) + H_{ii}(\tau)\pi_i \tag{59}$$

In theory, overshoot might happen at the end of a sprint, that is, θ_i might exceed θ_i^*. Nevertheless, overshoot cannot happen if parameter η is chosen large enough. θ_i can exceed θ_i^* after a reduction of θ_i^* but then the descent regime applies and θ_i decreases. Therefore, θ_i cannot exceed θ_{max}.

The descent regime is for enforcing SSPA. This can be understood as follows. Let us assume that the p_i^*'s are attainable. At steady-state, at least one of the following three conditions is true for each core i:

- $p_i = p_i^*$
- $\theta_i = \theta_i^*$
- $\theta_i < \theta_i^*$ and $p_i > p_i^*$

Indeed, the case $\theta_i < \theta_i^*$, $p_i < p_i^*$ means that core i is sprinting but power p_i^* is not attainable, which is contrary to our assumption. As for the case $\theta_i > \theta_i^*$, it cannot occur in steady state because the descent regime forces temperature to decrease. Hence we are left with the three cases listed above.

We prove by contradiction that we must have $\boldsymbol{p} = \boldsymbol{p^*}$. Assume $\boldsymbol{p} \neq \boldsymbol{p^*}$. As $\mathbf{R}$ is a positive definite matrix, we must have

$$(\boldsymbol{p} - \boldsymbol{p^*})^T \mathbf{R} (\boldsymbol{p} - \boldsymbol{p^*}) = (\boldsymbol{p} - \boldsymbol{p^*})^T (\boldsymbol{\theta} - \boldsymbol{\theta^*}) > 0$$

that is,

$$\sum_{i=1}^{N} (p_i - p_i^*)(\theta_i - \theta_i^*) > 0$$

However, the inequality above is incompatible with the three conditions previously mentioned. Therefore, we must have $\boldsymbol{p} = \boldsymbol{p^*}$. Note that it is important that $\mathbf{R}$ be well-conditioned as equality $\theta_i = \theta_i^*$ is only approximately true in practice.

Parameter ϵ determines how quickly temperature falls during a descent, as temperature follows an exponential decay with time constant τ/ϵ. Parameter ϵ does not need to be set very precisely. A reasonable value for ϵ is such that τ/ϵ is of the same order of magnitude as the time for a change of power in one core to start impacting temperature in a distant core.

One may argue that SSPA does not provide a true performance guarantee, in particular for applications whose execution time is too short for temperature to stabilize. Exploiting thermal transients while at the same time providing performance guarantees is a topic for further research and is beyond the scope of this document.

calculation	equation	scalar FP operation count
$\boldsymbol{X_j}$'s	(53)	$3Nn$
θ_i's	(54)	$(2N+1)N$
Δ_i's	(55)	$(2N+1)Nn$
$\boldsymbol{p} = \mathbf{B}^{-1}\boldsymbol{Y}$	(58)	$N(N-1)(4N+13)/6$

Table 5: Number of scalar FP operations per timestep of the MIMO turbo control (N is the number of cores and n the number of exponentials per impulse response).

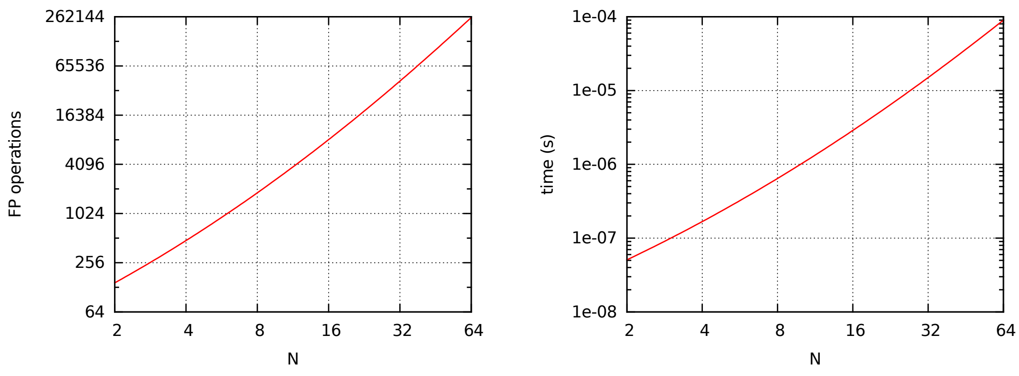


Figure 3: On the left, number of scalar FP operations per timestep of the MIMO turbo control, as a function of the number of cores N, for $n = 8$. On the right, corresponding CPU time (in seconds) on a 2.8 GHz Intel Sandy Bridge core.

Computational complexity. Calculations for the turbo control may be executed on dedicated hardware or by a firmware thread running on a core (hence consuming some CPU time).

Updating the $\boldsymbol{X_j}$'s requires $3Nn$ scalar floating-point (FP) operations (additions or multiplications) per timestep, assuming $e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}$ and $\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}$ have been pre-calculated (see equation (53)). The θ_i's are updated with equation (54), which requires $(2N+1)N$ scalar FP operations per timestep. The Δ_i's are calculated with equation (55). First, $\mathbf{I} - e^{-\mathbf{D}[\boldsymbol{\lambda}]\tau}$ is multiplied with the $\boldsymbol{X_j}$'s (Nn scalar multiplications). After that, $2N^2n$ operations are needed to obtain all Δ_i's, hence a total of $(2N+1)Nn$ FP scalar operations per timestep.

Equation (58) can be solved by Gaussian elimination.[18] This can be done with $2N-1$ divisions, $N(N-1)(N+4)/3$ multiplications and $N(N-1)(2N+5)/6$ additions. Neglecting divisions, that gives a total of about $N(N-1)(4N+13)/6$ scalar FP operations for solving (58).

Table 5 recapitulates FP operation counts, ignoring the operations for calculating $\boldsymbol{Y}$, $\tilde{\theta}_i(t+\tau)$, power from clock frequency, clock frequency from power, etc. The left-side graph of Figure 3 shows the number of scalar FP operations per timestep as a function of N, for $n = 8$. For small N, a majority of operations comes from calculating the Δ_i's and computational complexity increases roughly like N^2. As N increases, solving equation (58) takes more weight, and computational complexity approaches an N^3 growth.

I implemented the turbo control in C++ for the simulations presented in Section 8. I ran simulations on an Intel Sandy Bridge core with Turbo Boost disabled and maximum clock frequency 2.8 GHz. When compiled with gcc 4.9.2 using optimization level O3, the turbo control executes a number of x86 instructions about 2.65 times the number of scalar FP operations.[19] I

[18]Matrix $\mathbf{B}$ is column diagonally dominant, and no pivoting is needed.

[19]Gcc's auto-vectorization largely fails here. There is certainly room for improvement by manual vectorization.

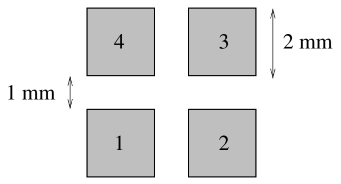


Figure 4: Example of Section 8. The four cores are identical. Each core is a 2mm×2mm square with uniform power density. A thermal sensor is located at the center of each core.

measured an IPC[20] of 2.7 on the Sandy Bridge core. The CPU time to execute the turbo control is

$$\text{CPU\_time} = \frac{\text{inst\_per\_FP} \times \text{FP\_operations}}{\text{IPC} \times \text{clock\_frequency}} = \frac{2.65 \times \text{FP\_operations}}{2.7 \times 2.8 \times 10^9}$$

where the number of FP operations is estimated from Table 5. The right-side graph of Figure 3 shows CPU time per timestep as a function of N, for $n = 8$. For example, assuming a timestep of one millisecond, the turbo control consumes less than 0.1% of the CPU time of one 2.8 GHz Sandy Bridge core when the number of cores is less than 10. To support a greater number of cores, we can consume more CPU time or increase the timestep, or both. For instance, if we are willing to consume 0.5% of the CPU time of one core and if use a two millisecond timestep, the turbo control can support up to 27 cores.

8 Example

This section uses the turbo control proposed in Section 7 in some high-level performance simulations of a 4-core chip ($N = 4$). The performance model is based on the following assumptions:

- Each of the 4 cores is a square dissipating a uniform power density, as shown in Figure 4.
- Each core has its own clock and voltage domain.
- Voltage can be changed instantaneously to any value.
- Normalized clock frequency can take any real value in $[0.5, 1]$.
- I used the affine power model (22), with $\alpha = -0.31$, $\beta = 1.18$, $\gamma = 0.0017$ and $P_w = 16$ W, where P_w is the maximum power of one core (the maximum power density is $P_w/4 = 4\,\text{W/mm}^2$). The power model is assumed quasi-perfect, in the sense that the actual power dissipated by a task is constant and equal to that predicted by the model except for the effect of temperature on leakage power (the power model uses θ_i, which may differ from the actual temperature, especially when the ambient temperature is not equal to 40 °C).
- The timestep of the turbo control is one millisecond ($\tau = 10^{-3}$ s). The clock frequency and power of each core are constant during a timestep.
- There is one thermal sensor at the center of each core. Sensors provide perfect temperature measurements at the end of a timestep, i.e., every millisecond (thermal protection uses the same timestep as the turbo control).

[20]Instructions executed per clock cycle.

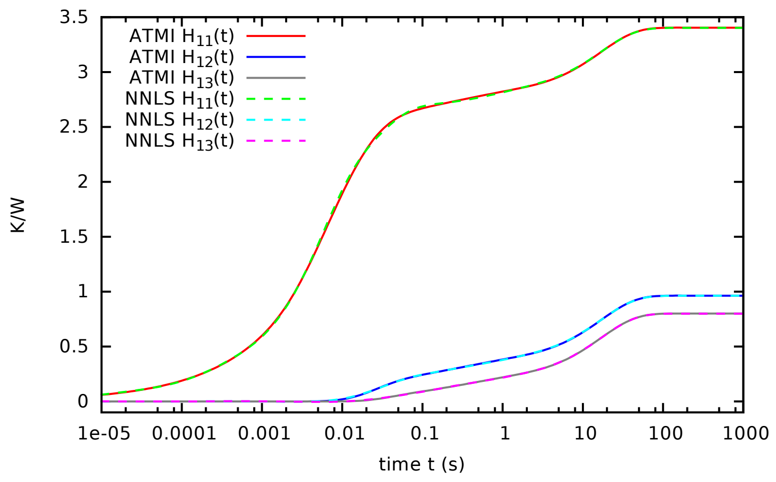


Figure 5: Example of Section 8: step responses $H_{11}(t)$, $H_{12}(t)$ and $H_{13}(t)$ obtained with ATMI, and approximations with Bernstein functions using the NNLS method ($n = 8$).

- Sensor temperature is modeled with ATMI [4]. ATMI is a linear model, i.e., the superposition principle (3) is exact [51]. The ATMI parameters are provided in Table 2. ATMI is also used to obtain the step responses for the turbo control.
- When any thermal sensor exceeds $T_{max} = 100$ °C, thermal protection triggers: the clock frequency of *all* the active cores is set to half the maximum clock frequency ($f = 0.5$ with the affine power model). The chip exits thermal protection when the triggering sensor drops below $T_{max} - 3 = 97$ °C.
- The IPC of a task does not depend on the clock frequency. Tasks execute a fixed number of instructions, hence a fixed number of clock ticks.
- The nominal ambient temperature is $T_{amb} = 40$ °C. The turbo control assumes a 1 °C guardband, that is, $\theta_{max} = T_{max} - T_{amb} - 1 = 59$ °C.
- MIMO turbo parameters are $\eta = 0.5$ and $\epsilon = 0.1$.
- For any active cores i and j, MIMO turbo enforces $p_i^* = p_j^*$.
- Power gating an inactive core or waking it up is instantaneous and generates no energy overhead. While inactive, a core dissipates a null power ($p_i^* = 0$).
- There is no constraint other than temperature in determining the clock frequency. In particular, $\boldsymbol{p^*}$ is set so that $\max\{\theta_1^*, \dots, \theta_N^*\} = \theta_{max}$.
- Calculations for the turbo control are done at no performance and energy cost.
- At $t = 0$, the chip, package and heatsink are at ambient temperature.

Some of these assumptions are obviously an idealization of reality. The goal of this section is not to obtain quantitatively accurate conclusions about real chips but to illustrate the proposed turbo control.

i	1	2	3	4	5	6	7	8
λ_i	0.0593	1.530	7.771	39.47	200.4	5169	26250	133300
$\boldsymbol{H_{11}}$	0.594	0.127	0.000	0.808	1.604	0.200	0.006	0.063
$\boldsymbol{H_{12}}$	0.599	0.093	0.084	0.273	-0.090	0.009	-0.006	0.002
$\boldsymbol{H_{13}}$	0.601	0.076	0.105	0.049	-0.036	0.009	-0.006	0.002

Table 6: Example of Section 8: step responses obtained with the NNLS method (λ_i's in s^{-1} and H_i's in K/W).

active cores	$\boldsymbol{p^*}$	$\boldsymbol{\theta^*}$
1	$(17.3, 0, 0, 0)$	$(59, 16.7, 13.9, 16.7)$
1,2	$(13.5, 13.5, 0, 0)$	$(59, 59, 23.8, 23.8)$
1,3	$(14, 0, 14, 0)$	$(59, 27, 59, 27)$
1,2,3	$(11.1, 11.1, 11.1, 0)$	$(57.2, 59, 57.2, 30.2)$
1,2,3,4	$(9.6, 9.6, 9.6, 9.6)$	$(59, 59, 59, 59)$

Table 7: Example of Section 8: $\boldsymbol{p^*}$ in watt and $\boldsymbol{\theta^*}$ in kelvin ($P_w = 16$ W, $\theta_{max} = 59$ K). Other task-to-core configurations (except the one with no active core) are similar to the configurations listed here, up to a permutation.

The step responses provided by ATMI are approximated with Bernstein functions, as described in sections 2.5 and 6. By trying a few values for n and λ_n and eliminating the λ_i's corresponding to null or quasi-null elements in all 16 $\boldsymbol{H_{ij}}$ vectors, I eventually obtained $n = 8$ λ_i's resulting in accurate approximations of the step responses, as shown in Figure 5. On the figure, the approximate step responses obtained with the NNLS method are barely distinguishable from those provided directly by ATMI. The λ_i's and $\boldsymbol{H_{ij}}$'s are given in Table 6. Only 3 reference step responses are shown: $H_{11}(t)$, $H_{12}(t)$ and $H_{13}(t)$. The 13 other step responses are identical to one of the 3 reference step responses owing to symmetries in the power density map of Figure 4 (ATMI does not model the edges of the silicon die).

Matrix $\mathbf{R}$ (in kelvin per watt) is

$$\mathbf{R} = \begin{pmatrix} 3.403 & 0.963 & 0.799 & 0.963 \\ 0.963 & 3.403 & 0.963 & 0.799 \\ 0.799 & 0.963 & 3.403 & 0.963 \\ 0.963 & 0.799 & 0.963 & 3.403 \end{pmatrix}$$

From matrix $\mathbf{R}$ and from the assumption of equal p_i^*'s for active cores, we obtain $\boldsymbol{p^*}$ and $\boldsymbol{\theta^*}$ as shown in Table 7. Notice that, when a single core is active, p_i^* is not attainable, as this requires $f > 1$. Notice also that, when three cores are active, the active cores do not behave identically, one core being hotter than the two other cores at equal power dissipation.

When p_i^* is attainable, the clock frequency corresponding to p_i^* can be obtained from (22): $f = (p_i^*/P_w - \alpha - \gamma\theta_{max})/\beta$. For one active core, $f = 1$ as p_i^* is no attainable. We have $f \simeq 0.89$ for two active cores ($p_i^* = 13.5$), $f \simeq 0.77$ for three active cores ($p_i^* = 11.1$) and $f \simeq 0.69$ for four active cores ($p_i^* = 9.6$). Nevertheless, these clock frequency values are for long running tasks and can be exceeded during thermal transients.

Figure 6 shows $\theta_i(t)$ and core power $p_i(t)$ for a simulation where fours tasks execute, one on each core. Each task executes a fixed number of instructions corresponding to 12 seconds of execution at maximum clock frequency. The tasks do not start at the same time but are staggered: task 1 starts executing at $t = 0$ on core 1, task 2 at $t = 3$ on core 2, task 3 at $t = 6$ on core 3, and task 4 at $t = 9$ on core 4. The following can be observed:

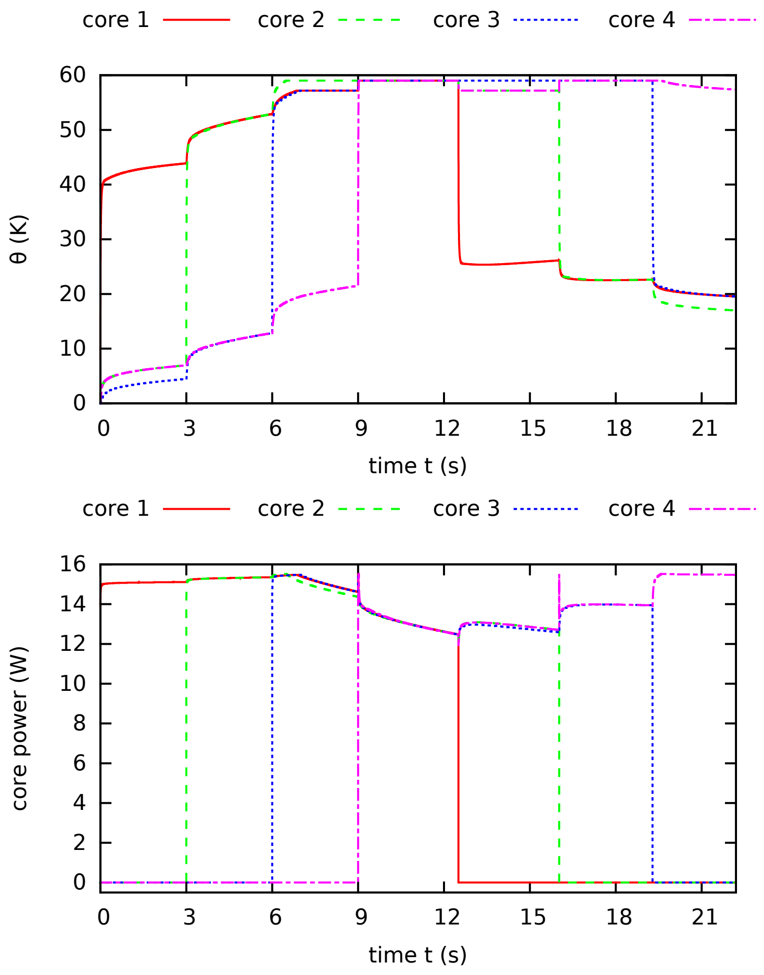


Figure 6: Example of Section 8: four tasks executing, one on each core. Each task executes a fixed number of instructions corresponding to 12 seconds of execution at maximum clock frequency. The first task starts executing at $t = 0$ on core 1, the second at $t = 3$ on core 2, the third at $t = 6$ on core 3, and the fourth at $t = 9$ on core 4. The upper graph shows calculated (relative) temperature $\theta_i(t)$. The lower graph shows core power $p_i(t)$. In this simulation, the actual ambient temperature is 40 °C and sensor-based thermal protection does not trigger.

- During the first 7 seconds, the active cores are sprinting and temperature increases. It can be noticed that the sprint power is slightly increasing due to the γ factor (leakage dependence on temperature).
- Core 2 becomes active at $t = 3$ and quickly reaches a temperature close to that of core 1. This can be explained as follows. After about 100 milliseconds of sprint, the X_i's corresponding to $\lambda_i \geq \lambda_4$ are all approximately equal to the sprint power, and therefore become quickly identical on both cores. The X_i's corresponding to $\lambda_i < \lambda_4$ contribute little to temperature except for X_1. However, as all the step responses have approximately the same $H_1 \simeq 0.6$ (see Table 6), the X_1 of a core generates the same temperature contribution on all cores.
- Cores 1, 2, and 3 reach thermal saturation during the time interval $[6, 9]$, after core 3 became active. Recall that cores 1 and 3 saturate at a slightly lower θ_i^* than core 2 (see Table 7).
- At $t = 9$, core 4 becomes active, and $\boldsymbol{p^*}$ and $\boldsymbol{\theta^*}$ are redefined. In particular, θ_1^* and θ_3^* increase from 57.2 to 59. Consequently, cores 1 and 3 can sprint for about 10 ms, which explains the power surge for these two cores at $t = 9$.
- At around $t = 12.5$, task 1 completes and only cores 2, 3 and 4 remain active. Consequently, θ_2^* and θ_4^* decrease from 59 to 57.2, and cores 2 and 4 quickly reduce their temperature under the descent regime (see Table 4).
- At around $t = 16$, task 2 completes and only cores 3 and 4 remain active. Consequently, θ_4^* increases from 57.2 to 59, hence the brief power surge on core 4.
- After task 3 completes around $t = 19.3$, only core 4 is active. Core 4 remains saturated for about 0.3 seconds, then its temperature decreases despite the core sprinting.

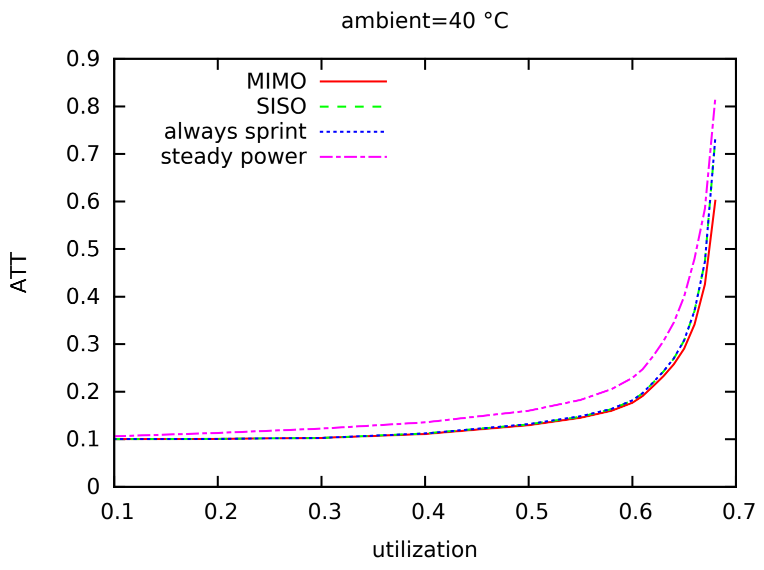
Then, I ran some simulations consisting of many short single-core tasks arriving in the system at random time, a typical model for cloud servers [48, 23]. Upon arrival, tasks are inserted into a global, unlimited first-in-first-out (FIFO) queue. When some cores are inactive and the queue is not empty, the task at the head of the queue (the oldest task) is removed and sent to execution on a core chosen randomly among the inactive cores. The task runs on that core until completion. Each task executes a fixed number of instructions corresponding to 0.1 seconds of execution at maximum clock frequency. The task interarrival times are exponentially distributed (Poisson process). The number of tasks in one simulation corresponds approximately to 300 seconds of execution (i.e., the shorter the interarrival times, the more jobs). Processor *utilization* and mean interarrival time are related as follows:

$$\text{utilization} := \frac{\text{task\_length}}{N \times \text{mean\_interarrival}} = \frac{0.1}{4 \times \text{mean\_interarrival}}$$

If the mean interarrival time exceeds $\frac{\text{task\_length}}{N \times f_s}$ where f_s is the clock frequency when all four cores are active and have reached thermal steady-state, then tasks arrive faster than the processor can execute in the long run, and the FIFO queue keeps growing. To avoid this situation, utilization is kept below $f_s = (p_i^*/P_w - \alpha - \gamma\theta_{max})/\beta \simeq 0.69$ (with $p_i^* = 9.6$, see Table 7).

Figure 7 shows the average turnaround time (ATT) for different policies. Turnaround time is the time from when a task arrives in the system until it is completed. Turnaround time includes the time spent waiting in the FIFO queue. Four different policies are being compared:

- The policy called "MIMO" is the MIMO turbo control described in Section 7.



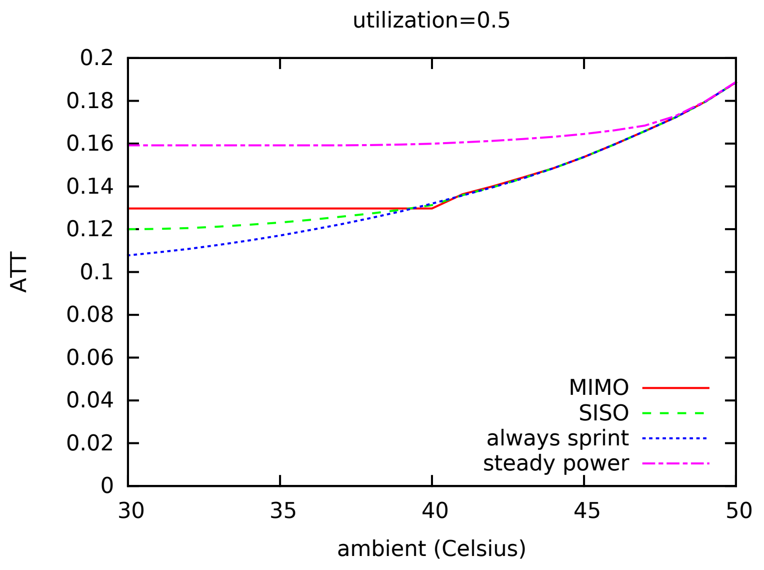


Figure 7: Average turnaround time (ATT) for 40 °C ambient and varying utilization (top graph), and for a processor utilization of 0.5 and varying ambient temperature (bottom graph).

- The policy called "SISO" is the SISO turbo control described in Section 4. The input of the SISO model is the total chip power $p = \sum_{i=1}^{4} p_i$. The step response is $H(t) = \frac{1}{4}\sum_{j=1}^{4} H_{1j}(t)$, which corresponds to the power density map of Figure 4 when each of the four cores dissipates 0.25 W. During thermal saturation, the calculated power is equally divided between the *active* cores (as tasks are identical, this implies that clock frequency is the same on all active cores).

- The policy called "always sprint" sets the clock frequency at the maximum value on all the active cores, unless sensor-based thermal protection is on. Note that this policy does not use a temperature model.

- The policy called "steady power" sets p_i equal to p_i^*, unless sensor-based thermal protection is on. Clock frequency is obtained from p_i assuming maximum temperature. Note that this policy models steady-state temperature, not transient temperature.

The top graph of Figure 7 shows ATT at an ambient temperature equal to the nominal ambient (40 °C), as a function of processor utilization. When utilization is low, the FIFO queue is empty most of the time, there is rarely more than one core active at a time, and tasks can execute at the maximum clock frequency. As a result, ATT is close to the task length (0.1 s). As utilization increases, so does the probability that the FIFO queue is not empty and that multiple cores are active simultaneously. Hence ATT increases with utilization. As expected, ATT becomes infinite when utilization approaches 0.69. Note that, at nominal ambient, the maximum utilization is the same for the four policies. Moreover, the MIMO, SISO and "always sprint" policies behave equivalently. However, ATTs are significantly higher with the "steady power" policy, which does not exploit thermal transients.

The bottom graph of Figure 7 shows ATT as a function of ambient temperature at a fixed processor utilization of 0.5. This graph illustrates the fact that, the more accurate the thermal model used in the turbo control, the more deterministic the processor performance. As expected (and helped by the ideal conditions of the simulation), performance with MIMO turbo is independent of the ambient temperature when below the nominal value. When ambient temperature exceeds the nominal value, the calculated temperature underestimates the actual temperature, and sensor-based thermal protection triggers sometimes, degrading performance. For ambient temperatures above 41 °C, the MIMO, SISO and "always sprint" policies behave similarly. However, with the "always sprint" policy, ATT keeps decreasing with the ambient temperature way below 40 °C. This is due to the "always sprint" policy relying solely on sensor-based thermal protection to control temperature. For instance, with the "always sprint" policy, at utilization of 0.5, thermal protection is active about 26% of the time at 40 °C ambient and 2% of the time at 30 °C ambient. Consequently, the "always sprint" policy outperforms MIMO turbo below the nominal ambient. However, the "always sprint" policy does not yield deterministic performance.[21] SISO turbo uses a simplistic temperature model which underestimates actual core temperature at nominal ambient. For instance, at utilization of 0.5 and 40 °C ambient, thermal protection is active 19% of the time with SISO turbo. Consequently, like the "always-sprint" policy, SISO turbo yields non-deterministic performance. However, performance dependence on ambient temperature is less pronounced with SISO turbo than with the "always sprint" policy.

[21] Besides making processor performance depend on ambient temperature, the "always sprint" policy does not offer SSPA.

9 Related work

Accurate electrothermal simulation of electronic circuits requires modeling the transient temperature of circuits with sufficient accuracy and simulation speed. This has led to a body of research on dynamic compact thermal models (DCTM) [67, 8].

One possible approach to DCTM, *system identification*, starts from a parametrized thermal model, power and temperature measurements of a real system, and tries to find parameters such that the model imitates the input-output behavior of the real system as closely as possible [9].

Another approach to DCTM, *model order reduction*, starts from an accurate thermal model based on spatial discretization, such as a finite-difference or finite-element model, and reduces the state-space dimension while preserving as much as possible the relations between (power) inputs and (temperature) outputs [42, 78, 20, 21, 18].

An oft-used DCTM, the RC ladder, approximates thermal step responses with equivalent Foster RC networks [42, 43, 26, 60, 57, 77]. The thermal model of section 2.6 is an RC ladder DCTM. Several methods can be used to extract the parameters of the multiexponential (9) [35]. For thermal step responses, commonly used methods are deconvolution by Fourier transform [76, 74], moment matching [42], Prony's method [43] and non-linear least-square fitting [26]. The non-negative least squares (NNLS) method has been used by some authors to extract the H_i's for self-heating step responses [7, 57]. Non-negativity constraints on the H_i's was justified either by one-dimensional heat conduction [6] or by assuming that a temperature step response is equivalent to an RC network impedance [75, 57] (see theorem 6.4 in [83]), which is not exact for general power density of the form (4) unless Codecasa's definition of output temperature (equation (12)) is used [19]. While Codecasa's definition leads to self-heating step responses that are provably Bernstein functions [17], the CMIR hypothesis as stated in Section 2.5 is a practical observation waiting for a proof.

In this document, the purpose of the thermal model is deterministic clock frequency. Fast thermal models are also needed for predictive dynamic thermal management (DTM). Predictive DTM is a general control strategy for minimizing an objective function while taking into account multiple constraints, such as maximum temperature, minimum clock frequency, voltage transition delay, etc. However, the thermal models used in predictive DTMs generally use thermal sensor information as input for calculating future temperature [81, 72, 30]. Unlike these predictive DTMs, the turbo control of Section 7 implements an open-loop control, i.e., without using thermal sensor information.

10 Conclusion

The turbo control proposed in this document considers junction temperature as the sole thermal constraint. However, in some handheld devices, case temperature is another constraint to take into account [63]. The thermal model of Section 6 can be adapted to output both junction and case temperatures (coplanarity is not an assumption of the model). For instance, an artificial "case" core, devoid of heat sources, can be added to the model. The maximum temperature in this "case" core would not be T_{max} but a value less than T_{max}. A specific turbo control would be needed however, as a "case" core is not exactly equivalent to an inactive core (whose temperature does not need to be monitored).

Figure 7 illustrates the fundamental tradeoff between maximizing clock frequency and making it deterministic. Clock frequency is more deterministic with the MIMO thermal model than with the SISO model, as the MIMO model is more accurate. For ambient temperatures below the nominal value T_{amb}, the SISO model yields higher clock frequencies. Above T_{amb}, the turbo control is generally superseded by the sensor-based DTM, and the thermal model is irrelevant.

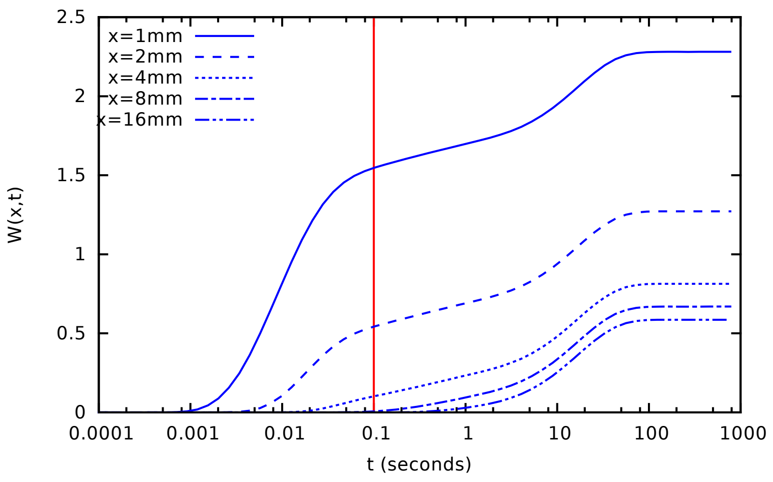


Figure 8: Step response $W(x,t)$ to a point source dissipating 1 watt, at distance x from the source, obtained with ATMI using the parameters of Table 2.

With the MIMO turbo, T_{amb} is the parameter separating precisely the deterministic behavior from the non-deterministic one. T_{amb} is included in the thermal model only via parameter θ_{max} and can easily be made configurable.

A Detailed power density is superfluous

If the point $\boldsymbol{r_\theta}$ where we are monitoring temperature $\theta(t)$ is not too close to a chip edge, the contribution to θ of a heat source located at a point $\boldsymbol{r}$ in the processor circuit is mostly a function of the distance $|\boldsymbol{r}-\boldsymbol{r_\theta}|$ between the two points. Therefore, to model temperature at $\boldsymbol{r_\theta}$, it is sufficient to know the amount of power within a certain distance from $\boldsymbol{r_\theta}$:

$$q(x,t) := \int_{|\boldsymbol{r}-\boldsymbol{r_\theta}|\leq x} g(\boldsymbol{r},t)d^3\boldsymbol{r}$$

From the superposition principle, we have

$$\theta(t) = \int_0^\infty \int_0^t q'(x,\tau)w(x,t-\tau)d\tau dx$$

where

$$q' := \frac{\partial q}{\partial x}$$

and where $w(x,t)$ is the time derivative of the temperature response $W(x,t) = u(\boldsymbol{r_\theta},t)$ to a point source at distance x from $\boldsymbol{r_\theta}$ dissipating the unit step power $U(t)$ (see Section 2.4), that is,

$$w = \frac{\partial W}{\partial t}$$

Figure 8 shows $W(x,t)$ for a few x values, obtained with ATMI using the parameters of Table 2. The greater the distance from the source, the more time it takes for heat to diffuse over that distance. For example, in Figure 8, it takes about 1 ms for heat to diffuse over 1 mm, and about 0.1 s to diffuse over 8 mm. Let us denote $x_0(t)$ the distance "traveled" by heat at time t, that is,

$$w(x,t) = W(x,t) = 0 \quad \text{for } x > x_0(t)$$

We have

$$\theta(t) = \int_0^t \int_0^{x_0(t-\tau)} q'(x,\tau) w(x,t-\tau) dx d\tau$$

It can also be observed in Figure 8 that, for $t > 0.1$ s, the curves $W(x,t)$ are almost parallel to each other. This can be explained as follows.

$w(x,t)$ is the temperature at $\boldsymbol{r_\theta}$ generated by the instantaneous release of one joule of energy at point $\boldsymbol{r}$ at time $t = 0$ in a system (chip + package + heatsink) that was at ambient temperature for $t < 0$. Just after the energy has been released, say for $0 < t < 1\,\mu$s, the vicinity of point $\boldsymbol{r}$ is very hot. Temperature in the rest of the system is still equal to the ambient, so the temperature gradients near $\boldsymbol{r}$ are very steep in radial directions, and heat quickly diffuses away from $\boldsymbol{r}$. Progressively, the hot region grows larger while at the same time becoming cooler, as the one joule of energy gets diluted in a larger volume. As the hot region becomes larger and cooler, it also becomes more thermally uniform under the effect of Fourier's law. Hence, after a certain time τ_0 (here, $\tau_0 \simeq 0.1$ s), point $\boldsymbol{r}$ is only marginally hotter than the rest of the hot region and $w(x,t)$ is only weakly dependent on x:

$$w(x,t) \simeq w_0(t) \quad \text{for } t > \tau_0 \text{ and } x < x_0(t)$$

Therefore,

$$\begin{aligned}\theta(t) &\simeq \int_0^{t-\tau_0} \int_0^{x_0(t-\tau)} q'(x,\tau) w_0(t-\tau) dx d\tau + \int_{t-\tau_0}^{t} \int_0^{x_0(t-\tau)} q'(x,\tau) w(x,t-\tau) dx d\tau \\ &\simeq \int_0^{t-\tau_0} q(x_0(t-\tau),\tau) w_0(t-\tau) d\tau + \int_{t-\tau_0}^{t} \int_0^{x_0(t-\tau)} q'(x,\tau) w(x,t-\tau) dx d\tau\end{aligned}$$

To model $\theta(t)$, it is sufficient to know

- $q(x,\tau)$ for $\tau \geq t - \tau_0$ and $x \leq x_0(t-\tau)$,
- $q(x_0(t-\tau),\tau)$ for $\tau < t - \tau_0$.

In conclusion, the spatial location of old power events does not need to be known precisely. What matters, for old power events, is the total chip power.

For example, it is generally more important to model θ accurately when it is high than when it is low. The total chip power must be modeled accurately even when processor activity is low, as this impacts future temperature. However, knowing the power density map precisely during periods of low activity is superfluous.

B Steady-state temperature

I assume that the heat transfer coefficient A is non-null on some parts of surface $\mathcal{S}$, so that a steady state is possible (otherwise, heat could not escape the system).

Let us consider constant power densities $g_1(\boldsymbol{r},t) = g_1(\boldsymbol{r})$ and $g_2(\boldsymbol{r},t) = g_2(\boldsymbol{r})$. The corresponding steady-state relative temperatures $u_1(\boldsymbol{r})$ and $u_2(\boldsymbol{r})$ are solutions of the steady-state heat equation

$$\boldsymbol{\nabla} \cdot \kappa \boldsymbol{\nabla} u + g = 0 \tag{60}$$

with boundary condition (see Section 2.2)

$$-\kappa \boldsymbol{\nabla} u \cdot \boldsymbol{n} = Au \tag{61}$$

We have

$$\begin{aligned}
\int_{\mathcal{V}} g_1 u_2 d^3\boldsymbol{r} &= -\int_{\mathcal{V}} u_2 \boldsymbol{\nabla} \cdot \kappa \boldsymbol{\nabla} u_1 d^3\boldsymbol{r} && \text{(62a)}\\
&= \int_{\mathcal{V}} \boldsymbol{\nabla} u_2 \cdot \kappa \boldsymbol{\nabla} u_1 d^3\boldsymbol{r} - \int_{\mathcal{V}} \boldsymbol{\nabla} \cdot (u_2 \kappa \boldsymbol{\nabla} u_1) d^3\boldsymbol{r} && \text{(62b)}\\
&= \int_{\mathcal{V}} \kappa \boldsymbol{\nabla} u_1 \cdot \boldsymbol{\nabla} u_2 d^3\boldsymbol{r} - \int_{\mathcal{S}} u_2 \kappa \boldsymbol{\nabla} u_1 \cdot \boldsymbol{n} d^2\boldsymbol{r} && \text{(62c)}\\
&= \int_{\mathcal{V}} \kappa \boldsymbol{\nabla} u_1 \cdot \boldsymbol{\nabla} u_2 d^3\boldsymbol{r} + \int_{\mathcal{S}} A u_1 u_2 d^2\boldsymbol{r} && \text{(62d)}
\end{aligned}$$

where we have used equation (60) in (62a), the divergence of scalar-vector product in (62b), the divergence theorem in (62c), and boundary condition (61) in (62d). Note that (62d) is symmetric with respect to u_1 and u_2, therefore a similar calculation yields

$$\int_{\mathcal{V}} g_2 u_1 d^3\boldsymbol{r} = \int_{\mathcal{V}} g_1 u_2 d^3\boldsymbol{r} \tag{63}$$

Also, for $g_1 = g_2 = g$, $u_1 = u_2 = u$, equation (62d) implies

$$\int_{\mathcal{V}} g u d^3\boldsymbol{r} \geq 0 \tag{64}$$

It should be noted that inequality (64) is true regardless of the sign of g and u (that g should be non-negative is a physical requirement, not a mathematical one). Moreover, for the integral in inequality (64) to be null, we must have $u = 0$ and $g = 0$ (from (62d) and from A being non-null on some parts of $\mathcal{S}$).

Let us consider a constant power density of the form

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

Power density map $m_j(\boldsymbol{r})$ generates steady temperature $u_j(\boldsymbol{r})$, and $g(\boldsymbol{r})$ generates temperature

$$u(\boldsymbol{r}) = \sum_{j=1}^{N} p_j u_j(\boldsymbol{r}) \tag{65}$$

Let us define average temperature as follows [19, 17]:

$$\bar{u}_i := \int_{\mathcal{V}} m_i(\boldsymbol{r}) u(\boldsymbol{r}) d^3\boldsymbol{r}$$

Introducing (65) into the definition above, we get

$$\bar{u}_i = \sum_{j=1}^{N} p_j \int_{\mathcal{V}} m_i(\boldsymbol{r}) u_j(\boldsymbol{r}) d^3\boldsymbol{r}$$

We can write the relation between the p_j's and the $\bar{u}_i$'s in matrix notation:

$$\bar{\boldsymbol{u}} = \bar{\mathbf{R}}\boldsymbol{p}$$

where $\bar{\boldsymbol{u}} = (\bar{u}_1, \dots, \bar{u}_N)$ and $\boldsymbol{p} = (p_1, \dots, p_N)$ are column vectors and $\bar{\mathbf{R}}$ is the $N \times N$ matrix such that

$$[\bar{\mathbf{R}}]_{ij} = \bar{R}_{ij} := \int_{\mathcal{V}} m_i(\boldsymbol{r}) u_j(\boldsymbol{r}) d^3\boldsymbol{r}$$

Note that, from (63), we have $\bar{R}_{ij} = \bar{R}_{ji}$ That is, matrix $\bar{\mathbf{R}}$ is symmetric.

Moreover,

$$\boldsymbol{p}^T\bar{\mathbf{R}}\boldsymbol{p} = \sum_{i=1}^{N} p_i\bar{u}_i = \sum_{i=1}^{N} p_i \int_{\mathcal{V}} m_i(\boldsymbol{r})u(\boldsymbol{r})d^3\boldsymbol{r} = \int_{\mathcal{V}} g(\boldsymbol{r})u(\boldsymbol{r})d^3\boldsymbol{r}$$

Hence inequality (64) implies $\boldsymbol{p}^T\bar{\mathbf{R}}\boldsymbol{p} \geq 0$.

If the power density maps $m_i(\boldsymbol{r})$ represent distinct cores, for each i, there exists an $\boldsymbol{r}_i$ such that $m_i(\boldsymbol{r}_i) \neq 0$ and $m_j(\boldsymbol{r}_i) = 0$ for all $j \neq i$. Therefore, $\boldsymbol{p} \neq \mathbf{0}$ implies $g \neq 0$ and $\boldsymbol{p}^T\bar{\mathbf{R}}\boldsymbol{p} > 0$. This means that matrix $\bar{\mathbf{R}}$ is *positive definite* [36]. As $\bar{\mathbf{R}}$ is symmetric and positive definite, its eigenvalues are real and positive [50].

Matrix $\mathbf{R}$ introduced in Section 6 can be viewed as a perturbation of matrix $\tilde{\mathbf{R}}$ defined as

$$\tilde{\mathbf{R}} := \bar{\mathbf{R}} + \mathbf{D}[d_1, \dots, d_N]$$

where $\mathbf{D}[d_1, \dots, d_N]$ is the diagonal matrix such that

$$d_i = R_{ii} - \bar{R}_{ii}$$

The difference between $\mathbf{R}$ and $\tilde{\mathbf{R}}$ is

$$\mathbf{E} := \mathbf{R} - \tilde{\mathbf{R}}$$

What makes $\mathbf{R}$ and $\bar{\mathbf{R}}$ differ from each other is that $\mathbf{R}$ gives maximum core temperature while $\bar{\mathbf{R}}$ gives average core temperature. So we have $d_i > 0$ for all i. Hence matrix $\tilde{\mathbf{R}}$ is symmetric, positive definite.

The elements of matrix $\mathbf{E}$ are expected to be small because there is probably not much difference between R_{ij} and $\bar{R}_{ij}$ when $i \neq j$.[22] If this is the case, $\mathbf{R}$ is close to $\tilde{\mathbf{R}}$, and $\mathbf{R}$ is well-conditioned if $\tilde{\mathbf{R}}$ is.

The condition number (according to the 2-norm, aka spectral norm) of a symmetric, positive definite matrix is the ratio of the largest to smallest eigenvalues [31, 50]. Here, the largest eigenvalue of $\tilde{\mathbf{R}}$ is the spectral radius $\rho(\tilde{\mathbf{R}}) \simeq \rho(\mathbf{R})$ and it is less than or equal to the maximum row sum (see Corollary 6.1.5 in [31]). That is, the largest eigenvalue does not exceed the maximum steady-state temperature when all cores dissipate 1 watt each.

A lower bound for the smallest eigenvalue can be obtained easily when $\mathbf{R}$ is *diagonally dominant* [50]. Here, diagonally dominant means $R_{ii} > \sum_{j,j\neq i} R_{ij}$ for all i. In this case, from Gerschgorin's theorem [50], the smallest eigenvalue of $\mathbf{R}$ cannot be less than

$$\min_i \Big\{ R_{ii} - \sum_{j,j\neq i} R_{ij} \Big\}$$

[22]Outside of core j, $\boldsymbol{\nabla} u_j$ in the silicon die is almost parallel to the transistors plane and u_j can be approximated by a harmonic function in two dimensions (as steady-state u_j obeys Laplace's equation in regions where $m_j = 0$). From the mean-value property [5], the average temperature in a disk-shaped region equals temperature at the center of the disk.

In the particular case of a chip with two *identical* cores ($N = 2$), the upper bound calculated as described above gives the exact condition number, that is, $\frac{R_{11}+R_{12}}{R_{11}-R_{12}}$. We can expect R_{12} to be significantly less than R_{11}, in which case $\mathbf{R}$ is well-conditioned.

When $N > 2$, $\mathbf{R}$ is not necessarily diagonally dominant. In this case, a lower bound for the smallest eigenvalue of $\tilde{\mathbf{R}}$ can be obtained from Weyl's inequality for Hermitian matrices [31]. That is, the smallest eigenvalue of $\tilde{\mathbf{R}}$ is greater than or equal to the sum of the smallest eigenvalues of $\bar{\mathbf{R}}$ and $\mathbf{D}[d_1, \ldots, d_N]$. As eigenvalues of $\bar{\mathbf{R}}$ are all positive, the smallest eigenvalue of $\tilde{\mathbf{R}}$ cannot be less than the smallest d_i.

For example, consider N identical cores occupying a fixed total silicon area. As N increases, the diameter of each core decreases like $\frac{1}{\sqrt{N}}$, and both R_{ii} and $\bar{R}_{ii}$ increase. More precisely, at small distances r from a point heat source, steady-state relative temperature varies approximately like $\frac{1}{r}$ [15, 52]. Therefore, R_{ii}, $\bar{R}_{ii}$, and their difference d_i increase like $\sqrt{N}$. As for the maximum steady-state temperature when all the cores dissipate 1 watt each, it increases approximately like total power, hence like N. Therefore, the upper bound for the condition number increases like $\frac{N}{\sqrt{N}} = \sqrt{N}$, that is, relatively slowly with the number of cores. This means that matrix $\mathbf{R}$ is probably well-conditioned.

References

[1] AMD. Advanced power management helps bring improved performance to highly integrated x86 processors. white paper, 2014. `https://www.amd.com/Documents/White_paper_layout_and_design.pdf`.

[2] AMD. Understanding power management and processor performance determinism. white paper, July 2018. `https://www.amd.com/system/files/documents/understanding-power-management.pdf`.

[3] V. S. Arpacı. *Conduction heat transfer.* Addison-Wesley, 1966.

[4] ATMI. Analytical model of temperature in microprocessors. `https://team.inria.fr/pacap/software/atmi`.

[5] S. Axler, P. Bourdon, and W. Ramey. *Harmonic function theory.* Springer, 1992.

[6] P. E. Bagnoli, C. Casarosa, M. Ciampi, and E. Dallago. Thermal resistance analysis by induced transient (TRAIT) method for power electronic devices thermal characterization - Part I: fundamentals and theory. *IEEE Transactions on Power Electronics*, 13(6), November 1998.

[7] P. E. Bagnoli, C. Casarosa, M. Ciampi, and E. Dallago. Thermal resistance analysis by induced transient (TRAIT) method for power electronic devices thermal characterization - Part II: practice and experiments. *IEEE Transactions on Power Electronics*, 13(6), November 1998.

[8] T. Bechtold, E. B. Rudnyi, and J. G. Korvink. Dynamic electro-thermal simulation of microsystems - a review. *Journal of Micromechanics and Microengineering*, 15(11), October 2005.

[9] F. Beneventi, A. Bartolini, A. Tilli, and L. Benini. An effective gray-box identification procedure for multicore thermal modeling. *IEEE Transactions on Computers*, 63(5), May 2014.

[10] G. Blake, R. G. Dreslinski, T. Mudge, and K. Flautner. Evolution of thread-level parallelism in desktop applications. In *International Symposium on Computer Architecture (ISCA)*, 2010.

[11] B. Bowhill, B. Stackhouse, N. Nassif, Z. Yang, A. Raghavan, O. Mendoza, C. Morganti, C. Houghton, D. Krueger, O. Franza, J. Desai, J. Crop, B. Brock, D. Bradley, C. Bostak, S. Bhimji, and M. Becker. The Xeon processor E5-2600 v3: a 22nm 18-core product family. *IEEE Journal of Solid-State Circuits*, 51(1), January 2016.

[12] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *International Symposium on High-Performance Computer Architecture (HPCA)*, 2001.

[13] T. Burd, N. Beck, S. White, M. Paraschou, N. Kalyanasundharam, G. Donley, and A. Smith. Zeppelin: an SoC for multichip architectures. *IEEE Journal of Solid-State Circuits*, 54(1), January 2019.

[14] L. Cao, J. P. Krusius, M. Korhonen, and T. Fisher. Transient energy management strategies for portable systems. In *Electronic Components and Technology Conference (ECTC)*, 1995.

[15] H. S. Carslaw and J. C Jaeger. *Conduction of heat in solids*. Oxford University Press, second edition, 1959.

[16] C.-T. Chen. *Linear System Theory and Design*. Oxford University Press, third edition, 1999.

[17] L. Codecasa. Canonical forms of one-port passive distributed thermal networks. *IEEE Transactions on Components and Packaging Technologies*, 28(1), March 2005.

[18] L. Codecasa. Compact models of dynamic thermal networks with many heat sources. *IEEE Transactions on Components and Packaging Technologies*, 30(4), December 2007.

[19] L. Codecasa, D. D'Amore, and P. Maffezzoni. A rigorous approach to electro-thermal network modeling. In *European Conference on Circuit Theory and Design (ECCTD)*, 2001.

[20] L. Codecasa, D. D'Amore, and P. Maffezzoni. An Arnoldi based thermal network reduction method for electro-thermal analysis. *IEEE Transactions on Components and Packaging Technologies*, 26(1), March 2003.

[21] L. Codecasa, D. D'Amore, and P. Maffezzoni. Compact modeling of electrical devices for electrothermal analysis. *IEEE Transactions on Circuits and Systems I*, 50(4), April 2003.

[22] A. Cohen, L. Finkelstein, A. Mendelson, R. Ronen, and D. Rudoy. On estimating optimal performance of a CPU dynamic thermal management. *IEEE Computer Architecture Letters*, 2(1), October 2003.

[23] C. Delimitrou and C. Kozyrakis. Amdahl's law for tail latency. *Communications of the ACM*, 61(8), August 2018.

[24] J. Doweck, W.-F. Kao, A. K. Lu, J. Mandelblat, A. Rahatekar, L. Rappoport, E. Rotem, A. Yasin, and A. Yoaz. Inside 6th-generation Intel Core: new microarchitecture code-named Skylake. *IEEE Micro*, 37(2), March 2017.

[25] L. Emurian, A. Raghavan, L. Shao, J. M. Rosen, M. Papaefthymiou, K. Pipe, T. F. Wenisch, and M. Martin. Pitfalls of accurately benchmarking thermally adaptive chips. In *Workshop on Duplicating, Deconstructing and Debunking (WDDD)*, 2014.

[26] Y. C. Gerstenmaier and G. Wachutka. Calculation of the temperature development in electronic systems by convolution integrals. In *Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM)*, 2000.

[27] Y. C. Gerstenmaier and G. Wachutka. Time dependent temperature fields calculated using eigenfunctions and eigenvalues of the heat conduction equation. *Microelectronics Journal*, 32(10), October 2001.

[28] B. Goel and S. A. McKee. A methodology for modeling dynamic and static power consumption for multicore processors. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2016.

[29] G. Gripenberg, S.-O. Londen, and O. Staffans. *Volterra integral and functional equations.* Cambridge University Press, 1990.

[30] V. Hanumaiah, D. Desai, B. Gaudette, C.-J. Wu, and S. Vrudhula. STEAM: a smart temperature and energy aware multicore controller. *ACM Transactions on Embedded Computing Systems*, 13(5), September 2014.

[31] R. A. Horn and C. R. Johnson. *Matrix analysis.* Cambridge University Press, second edition, 2013.

[32] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan. HotSpot: a compact thermal modeling methodology for early-stage VLSI design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5), May 2006.

[33] W. Huang, C. Lefurgy, W. Kuk, A. Buyuktosunoglu, M. Floyd, K. Rajamani, M. Allen-Ware, and B. Brock. Accurate fine-grained processor power proxies. In *International Symposium on Microarchitecture (MICRO)*, 2012.

[34] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: methodology and empirical data. In *International Symposium on Microarchitecture (MICRO)*, 2003.

[35] A. A. Istratov and O. F. Vyvenko. Exponential analysis in physical phenomena. *Review of Scientific Instruments*, 70(2), February 1999.

[36] C. R. Johnson. Positive definite matrices. *The American Mathematical Monthly*, 77(3), 1970.

[37] H. Kasture, D. B. Bartolini, N. Beckmann, and D. Sanchez. Rubik: fast analytical power management for latency-critical systems. In *International Symposium on Microarchitecture (MICRO)*, 2015.

[38] J. Keilson. Exponential spectra as a tool for the study of server-systems with several classes of customers. *Journal of Applied Probability*, 15(1), March 1978.

[39] D. Kim, S. Sra, and I. S. Dhillon. A non-monotonic method for large-scale non-negative least squares. *Optimization Methods and Software*, 28(5), 2013.

[40] D. Lacroix, K. Joulain, and D. Lemonnier. Monte Carlo transient phonons transport in silicon and germanium at nanoscales. *Physical Review B*, 72(6), August 2005.

[41] C. L. Lawson and R. J. Hanson. *Solving least squares problems.* Society for Industrial and Applied Mathematics, 1995.

[42] S.-S. Lee and D. J. Allstot. Electrothermal simulation of integrated circuits. *IEEE Journal of Solid-State Circuits*, 28(12), 1993.

[43] T. Li, C.-H. Tsai, and S.-M. Kang. Efficient transient electrothermal simulation of CMOS VLSI circuits under electrical overstress. In *International Conference on Computer-Aided Design (ICCAD)*, 1998.

[44] W. Liao, L. He, and K. M. Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(7), July 2005.

[45] P. Linz. *Analytical and numerical methods for Volterra equations.* Society for Industrial and Applied Mathematics, 1985.

[46] D. Lo and C. Kozyrakis. Dynamic management of TurboMode in modern multi-core chips. In *International Symposium on High Performance Computer Architecture (HPCA)*, 2014.

[47] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: eliminating server idle power. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2009.

[48] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch. Power management of online data-intensive services. In *International Symposium on Computer Architecture (ISCA)*, 2011.

[49] M. Merkle. Completely monotone functions: a digest. In G. V. Milovanović and M. T. Rassias, editors, *Analytic Number Theory, Approximation Theory, and Special Functions.* Springer, 2014.

[50] C. D. Meyer. *Matrix analysis and applied linear algebra.* Society for Industrial and Applied Mathematics, 2000.

[51] P. Michaud and Y. Sazeides. ATMI: analytical model of temperature in microprocessors. In *Workshop on Modeling, Benchmarking and Simulation*, 2007.

[52] P. Michaud, Y. Sazeides, A. Seznec, T. Constantinou, and D. Fetis. An analytical model of temperature in microprocessors. Technical Report RR-5744, Inria, 2005.

[53] NNLS. Super fast large-scale nonnegative least squares. `http://suvrit.de/work/soft/nnls.html`.

[54] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. *Signals and Systems.* Prentice Hall, second edition, 1997.

[55] D. Orenstien and R. Ronen. Deterministic power-estimation for thermal control. US patent 7096145, January 2002.

[56] M.N. Özişik. *Boundary value problems of heat conduction.* Dover Publications, 1968.

[57] J. Palacín, M. Salleras, J. Samitier, and S. Marco. Dynamic compact thermal models with multiple power sources: application to an ultrathin chip stacking technology. *IEEE Transactions on Advanced Packaging*, 28(4), November 2005.

[58] S. Park, J. Park, D. Shin, Y. Wang, Q. Xie, M. Pedram, and N. Chang. Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(5), May 2013.

[59] R. Rao, S. Vrudhula, C. Chakrabarti, and N. Chang. An optimal analytical solution for processor speed control with thermal constraints. In *International Symposium on Low Power Electronics and Design (ISLPED)*, 2006.

[60] M. Rencz, V. Székely, and A. Poppe. A fast algorithm for the layout based electro-thermal simulation. In *Design, Automation and Test in Europe Conference (DATE)*, 2003.

[61] E. Rohou and M. D. Smith. Dynamically managing processor temperature and power. In *Workshop on Feedback-Directed Optimization*, 1999.

[62] E. Rotem. Intel architecture, codename Skylake deep dive: a new architecture to manage power performance and energy efficiency. Intel Developer Forum, 2015. `https://en.wikichip.org/wiki/File:Intel_Architecture,_Code_Name_Skylake_Deep_Dive-_A_New_Architecture_to_Manage_Power_Performance_and_Energy_Efficiency.pdf`.

[63] 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.

[64] 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`.

[65] 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.

[66] E. Rotem, D. Rajwan, and L. Finkelstein. Deterministic management of dynamic thermal response of processors. US patent 8707060, October 2008.

[67] 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.

[68] E. C. Samson, S. V. Machiroutu, J.-Y. Chang, I. Santos, J. Hermerding, A. Dani, R. Prasher, and D. W. Song. Interface material selection and a thermal management technique in second-generation platforms built on Intel Centrino Mobile Technology. *Intel Technology Journal*, 9(1), 2005.

[69] H. Sanchez, B. Kuttanna, T. Olson, M. Alexander, G. Gerosa, R. Philip, and J. Alvarez. Thermal management system for high performance PowerPC microprocessors. In *IEEE COMPCON*, 1997.

[70] R. L. Schilling, R. Song, and Z. Vondraček. *Bernstein functions*. De Gruyter, 2010.

[71] E. Seneta. *Non-negative matrices and Markov chains*. Springer, second edition, 1981.

[72] S. Sharifi, D. Krishnaswamy, and T. Šimunić Rosing. PROMETHEUS: a proactive method for thermal management of heterogeneous MPSoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(7), July 2013.

[73] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *International Symposium on High-Performance Computer Architecture (HPCA)*, 2002.

[74] V. Székely. Identification of RC networks by deconvolution: chances and limits. *IEEE Transactions on Circuits and Systems I*, 45(3), March 1998.

[75] V. Székely and M. Rencz. Thermal dynamics and the time constant domain. *IEEE Transactions on Components and Packaging Technologies*, 23(3), September 2000.

[76] V. Székely and T. Van Bien. Fine structure of heat flow path in semiconductor devices: a measurement and identification method. *Solid-State Electronics*, 31(9), September 1988.

[77] M. N. Touzelbaev, J. Miler, Y. Yang, G. Refai-Ahmed, and K. E. Goodson. High-efficiency transient temperature calculations for applications in dynamic thermal management of electronic devices. *ASME Journal of Electronic Packaging*, 135(3), September 2013.

[78] C.-H. Tsai and S.-M. Kang. Fast temperature calculation for transient electrothermal simulation by mixed frequency/time domain thermal model reduction. In *Design Automation Conference (DAC)*, 2000.

[79] H. Vandierendonck and A. Seznec. Fairness metrics for multi-threaded processors. *IEEE Computer Architecture Letters*, 10(1), 2011.

[80] A. Vassighi and M. Sachdev. Thermal runaway in integrated circuits. *IEEE Transactions on Device and Materials Reliability*, 6(2), June 2006.

[81] Y. Wang, K. Ma, and X. Wang. Temperature-constrained power control for chip multiprocessors with online model estimation. In *International Symposium on Computer Architecture (ISCA)*, 2009.

[82] N. H. E. Weste and D. M. Harris. *CMOS VLSI design: a circuits and systems perspective.* Addison-Wesley, fourth edition, 2010.

[83] O. Wing. *Classical circuit theory.* Springer, 2009.

[84] S. M. Zemyan. *The classical theory of integral equations.* Birkhäuser, 2012.

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399