3D-ICE 3.0: Efficient Nonlinear MPSoC Thermal Simulation With Pluggable Heat Sink Models

Federico Terraneo, Alberto Leva, *Member, IEEE*, William Fornaciari, *Senior Member, IEEE*, Marina Zapater, *Member, IEEE*, and David Atienza, *Fellow, IEEE*

***Abstract*—The increasing power density in modern high-performance multiprocessor System-on-Chip (MPSoC) is fueling a revolution in thermal management. On the one hand, thermal phenomena are becoming a critical concern, making accurate and efficient simulation a necessity. On the other hand, a variety of physically heterogeneous solutions is coming into play: liquid, evaporative, thermoelectric cooling, and more. A new generation of simulators, with unprecedented flexibility, is thus required. In this article, we present 3D-ICE 3.0, the first thermal simulator to allow for accurate nonlinear descriptions of complex and physically heterogeneous heat dissipation systems, while preserving the efficiency of latest compact modeling frameworks at the silicon die level. 3D-ICE 3.0 allows designers to extend the thermal simulator with new heat sink models while simplifying the time-consuming step of model validation. The support for nonlinear dynamic models is included, for instance, to accurately represent variable coolant flows. Our results present validated models of a commercial water heat sink and an air heat sink plus fan that achieve an average error below 1 °C and simulate, respectively, up to 3× and 12× faster than the real physical phenomena.**

***Index Terms*—Computer architecture, co-simulation, nonlinear systems simulation, power management, thermal management, thermal simulation.**

I. Introduction

The RACE toward increasing performance is constantly pushing the power density of integrated circuits. The thermal design power (TDP) of high performance multiprocessor System-on-Chips (MPSoCs) has recently reached 400 W [1] and is expected to further increase [2], [3]. Heat dissipation saw a first revolution in the Pentium era with the introduction of fan cooling. The second one is now taking place as liquid cooling makes its way in high-end desktop PCs [4] and datacenters, where it is rapidly gaining importance [5]. Other cooling technologies on the horizon are evaporative cooling [6], combined cooling and on-chip power generation [7], thermoelectric cooling [8], and more [9].

As computer architectures are becoming increasingly heterogeneous [10], the same is happening for heat dissipation solutions. This trend, coupled to the already mentioned TDP boost, will make the design and management of such new solutions critical for the life itself of MPSoCs [11]. Hence, thermal simulators play a key role in this scenario. They allow to effectively and economically evaluate different dissipation solutions, as well as to perform design space explorations and to aid the design of thermal control policies. However, existing state-of-the-art thermal simulators [12]–[14] are bound to a limited number of simple dissipation solutions, often just a sink with constant fan speed. These current simulators can be configured to change the chip and sink sizes, materials, and heat transfer coefficient toward the ambient, however, it is not possible, for example, to represent an arbitrary sink shape or the effect of variable fan speed, let alone addressing more complicated solutions, such as liquid cooling circuits.

To understand the fundamental reason for this limitation, we must consider that to model heat transfer phenomena, thermal simulators have to solve differential equations [15] of a nature that changes dramatically across heat dissipation solutions. Moreover, to improve performance, current simulators adopt a monolithic design and essentially hardcode the equations describing the physics of the problem. This approach is acceptable for the chip level part of the simulator, at least as long as no on-chip cooling is present, since in this case, the chip can be invariantly described as the problem of heat conduction in a solid. Conversely, the same approach becomes a significant barrier for describing the diverse phenomena occurring in modern heat dissipation solutions.

Overcoming the previously mentioned limitation requires, thus, a different design perspective. Thermal simulators must be open to *extensibility*. In addition, to describe the physics of advanced cooling solutions as well as variable coolant flow in air and liquid cooling, *nonlinear* differential equations are required. As will be shown later on, the monolithic design approach is not well suited for this purpose.

In this work, we introduce 3D-ICE 3.0, a thermal simulator designed to address the aforementioned challenges. 3D-ICE 3.0 builds upon the 3D-ICE [12] thermal simulator,

Manuscript received September 14, 2020; revised December 11, 2020 and February 18, 2021; accepted April 13, 2021. Date of publication April 20, 2021; date of current version March 21, 2022. This work was supported in part by the EC H2020 RECIPE Project under Grant 801137, and in part by the ERC Consolidator Grant COMPUSAPIEN under Grant 725657. The work of Federico Terraneo was supported by the EC EUROLAB-4-HPC CSA Grant through a Short Term Collaboration Project with EPFL under Grant 800962. This article was recommended by Associate Editor W. Yu. *(Corresponding author: Federico Terraneo.)*

Federico Terraneo, Alberto Leva, and William Fornaciari are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy (e-mail: federico.terraneo@polimi.it; alberto.leva@polimi.it; william.fornaciari@polimi.it).

Marina Zapater is with the School of Management and Engineering Vaud (HEIG-VD), University of Applied Sciences Western Switzerland (HES-SO), 1401 Yverdon-les-Bains, Switzerland (e-mail: marina.zapater@heig-vd.ch).

David Atienza is with the Embedded Systems Laboratory, Swiss Federal Institute of Technology Lausanne (EPFL), 1015 Lausanne, Switzerland (e-mail: david.atienza@epfl.ch).

Digital Object Identifier 10.1109/TCAD.2021.3074613

by extending the differential equation model of the silicon die with a co-simulation interface to connect it with arbitrary heat sink models. This approach has the added benefit of partitioning the model validation issue, as no modification is needed to the core part of the simulator to add a new heat sink. Our purpose is that 3D-ICE 3.0 enables the computer and systems engineers to test different cooling techniques without the need of performing any validation; while at the same time, enabling thermal experts to simulate novel cooling solutions without the need to revalidate chip models.

The proposed co-simulation interface follows the functional mockup interface (FMI) industry standard [16], allowing to integrate a vast set of languages and tools, including equation-based ones, such as Modelica and Simscape [17]. With such languages, it is possible to create a heat sink model directly in terms of its equations, as opposed to writing the code to solve them, which is much faster for the MPSoC designer. These languages also natively support the efficient solution of nonlinear differential equations. Opening MPSoC simulation to the equation-based modeling paradigm is, thus, a key feature of our simulator, as it provides access to powerful technologies for addressing the heterogeneous heat sinks modeling challenge.

To show the potential of 3D-ICE 3.0, we present new validated models for a commercial air heat sink with variable fan speed, and a commercial water cooled one with variable water flow rate. These models can be used to perform architectural exploration as well as to design new thermal policies for different MPSoC architectures. More specifically, this article introduces the following main contributions.

1) We present a new thermal simulator with a very flexible interface for co-simulation. Thus, we enable the simulation of arbitrary cooling models without the need to change the underlying chip thermal model.
2) Concerning heat sink simulation capabilities, we support for the first time heat sink models including *nonlinear differential equations*. Therefore, we make possible to simulate physical phenomena very rapidly, which cannot be simulated with existing thermal simulators.
3) We simplify the design of heat sinks for MPSoC designers by allowing to express them directly as equations instead of code. Moreover, we provide 3D-ICE 3.0 as an open-source code library.[1] Consequently, we encourage the contribution of new heat sink models, in order to create an ecosystem of interoperable models around the 3D-ICE 3.0 core, addressing the need for heterogeneous heat sink modeling for next-generation MPSoCs.
4) We introduce *two new heat sink models* to address air and water cooling with variable coolant flow rate support. The air heat sink model can simulate up to 12 times faster than the real physical phenomena (i.e., faster than real time), while the water heat sink is up to three times faster than real time. Both have been validated against experimental results, achieving an average error below 1 °C.

The remainder of this article is organized as follows. Section II provides an overview of existing thermal simulators. Section III describes from a theoretical perspective the approach to thermal modeling adopted by 3D-ICE 3.0, while Section IV details the key implementation considerations. Section V introduces our new heat sink models. Sections VI–VIII present experimental results and discusses the validation of the proposed simulator and heat sink models. Finally, Section IX summarizes the main conclusions of this work.

II. Related Work

MPSoCs thermal simulators started appearing when the power and thermal wall became relevant for computer architectures, to perform thermal-aware MPSoC design and evaluate thermal control policies [12], [13].

HotSpot [13] was the first thermal simulator to reach wide adoption and is still one of the most used, thanks to its ease of integration in an MPSoC simulation workflow. HotSpot supports 2-D and 3-D integrated circuits and a conventional air-cooling heat sink. It has a monolithic architecture where both the silicon and heat sink are simulated in a tightly coupled codebase. The solver used is a fourth-order Runge Kutta (RK4) [24], and can only solve linear equation systems.

The ISAC [18] simulator introduced an adaptive spatial grid and an adaptive simulation time step. Its major feature is improved simulation performance, claiming above 10× speedup compared to previous methods. Nonetheless, its capabilities are similar to HotSpot, as it uses the same integration method, and is a strictly linear monolithic simulator.

LUTSim [19] improves the simulation speed and allows incremental simulation by adopting a look-up table approach, claiming a 39× speedup compared to the SuperLU linear algebra library used in thermal simulators such as 3D-ICE. However, the approach is monolithic and its support for heat sink models is very basic. Significant speedups have been obtained while considering heat sinks in greater detail through domain decomposition methods [20]; however, both approaches are limited to linear systems and only support steady state simulation.

Next, 3D-ICE [12] extended the capabilities of MPSoC thermal simulators by adding support for on-chip liquid cooling through microchannels etched in the silicon. 3D-ICE supports linear differential equations and solves them using the implicit Euler integration method [12]. Therefore, liquid cooling support has been added by linearizing the equations of convective heat transfer, and adding the linearized equation to the ones modeling the silicon in a single monolithic codebase. Liquid cooling is limited to constant water flow rates.

Although most MPSoC thermal simulators solve the partial differential equations of heat transfer using the finite volumes method, Ladenheim *et al.* [21] proposed the finite elements one to improve accuracy. The quoted simulator is, however, monolithic and only supports conventional heat dissipation solutions, which are modeled using strictly linear equations.

From the literature reviewed so far, it can be seen that most of the advancements in thermal simulation beyond 3D-ICE is directed either toward improving simulation speed, accuracy,

[1] https://www.epfl.ch/labs/esl/research/open-source-software-projects/3d-ice

TABLE I
COMPARISON OF THERMAL SIMULATORS CAPABILITIES

Simulator	Differential equation solver				Air cooling		Water cooling	
					Constant	Variable	Constant	Variable
	Integration	Transient	Nonlinear	Co-simulation	flow	flow	flow	flow
HotSpot [13]	RK4	✓			✓			
3D-ICE [12]	Euler	✓			Basic		✓	
ISAC [18]	RK4	✓			Basic			
LUTSim [19]	n/a[1]				Basic			
Yu *et al.* [20]	n/a[1]				✓			
Ladenheim *et al.* [21]	Krylov	✓			✓			
LoCool [9]	n/a[1]				✓		✓	
Therminator [22]	n/a[1]				✓			
NanoHeat [23]	NHAR	✓			✓			
MTA [14]	Krylov	✓	✓		✓			
This work	Euler	✓	✓	✓	✓	✓	✓	✓

[1] Steady-state only, no solution of differential equations.

or a combination thereof. On the other hand, the improvement of the range of heat dissipation solutions that can be simulated received far more limited attention, especially when considering transient simulation support. Among works that are the nearest to our design goals, we can cite LoCool [9], a recent thermal model that extends HotSpot with a compact model for on-chip liquid cooling similar to the 3D-ICE one, as well as a linearized thermoelectric cooler (TEC) model to model localized hotspot cooling. Although this simulator adds new cooling capabilities, its monolithic architecture still does not make it open to extensibility toward generic cooling methods. Moreover, the approach is monolithic and limited to steady-state simulations.

Therminator [22] is a thermal simulator for smartphones. Its most impressive feature is the capability to perform a thermal simulation of an entire phone, including the MPSoC, printed circuit, battery, case, and screen. However, the simulation is monolithic and linear in nature, and limited to steady state. Moreover, it does not support different cooling systems.

Also, NanoHeat [23] brings ISAC's adaptive spatial grid to the next level by bringing simulation down to gate resolution by accounting for phonon effects. Then, to make the problem tractable, the nanoscale thermal simulation is computed offline on each entry of a technology library. This approach produces look-up tables that are superimposed to a conventional macroscale thermal simulation. Although this simulator allows to see temperature variations at a fine level of detail, the baseline temperatures over which the detail is applied still depend on an accurate simulation of the heat sink. Furthermore, its heat sink support is still monolithic and linear.

An interesting alternative is MTA [14], which introduces support for nonlinear differential equations—though limited to modeling the temperature-dependent material properties inside the chip due to the monolithic approach. MTA, thus, addresses an issue that is relevant in the presence of large silicon temperature variations—hence, a different problem with respect to ours—and takes the right approach to that.

Table I summarizes the main supported features of state-of-the-art thermal simulators. Differential equation capabilities determine the type and variety of models that can be simulated. Most simulators support transient simulation, which is important to observe the large temporal temperature variations in modern MPSoCs [25]. Indeed, support for nonlinear differential equations throughout the chip and sink compound future-proofs a simulator, as it decides on the possibility of modeling innovative heat sink solutions; it is, however, an uncommon feature, in general, not yet fully exploited. Our proposal with 3D-ICE 3.0 addresses a part of the problem that we deem particularly relevant, namely, adding nonlinear support for the heat dissipation system while leaving the chip core simulator untouched. 3D-ICE 3.0 is also the only MPSoC thermal simulator supporting co-simulation, which lowers the barrier to prototyping and designing heat sinks, while also partitioning the validation burden.

As for the heat sink support included in current MPSoC simulators, some only support basic convective air cooling, modeled in terms of a heat transfer coefficient. More advanced simulators also exist that model the heat capacity of the sink material and the fin structure, while only 3D-ICE supports fixed coolant flow rate on-chip water cooling. 3D-ICE 3.0 improves support further by providing validated models of air and water cooling heat sinks capable of operating up to 12× faster than real time, while having an average error below 1 °C, despite still supporting variable coolant flow rates to the benefit of thermal policies.

III. 3D-ICE 3.0: PROPOSED METHODOLOGY

In this section, we illustrate the 3D-ICE 3.0 approach and its motivations, analyzing the thermal modeling problem in MPSoCs from a mathematical viewpoint. The goal is to highlight the difficulties entailed by modern dissipation systems, and the consequences of these on the required simulation techniques.

A. Modeling of Heating Solids With Conventional Dissipation

To model a simple solid body (e.g., a cube) that releases heat to an external environment at prescribed conditions, the necessary mathematical tools are the 3-D Fourier equation and a convective exchange correlation [15], [26]. In the three spatial coordinates (x, y, z) $[m]$, the former reads

$$\frac{\partial T(x, y, z, t)}{\partial t} = \frac{\lambda(x, y, z)\nabla^2 T(x, y, z, t) + Q(x, y, z, t)}{c(x, y, z)\rho(x, y, z)} \quad (1)$$

where $T(x, y, z, t)$ is the temperature $[K]$, as function of space and time, $Q(x, y, z, t)$ is the internal volume heat generation rate [W/m^3], ∇^2 is the Laplacian operator

$$\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \tag{2}$$

and ρ, c, and λ are the material density [kg/m^3], specific heat [J/kgK], and thermal conductivity [W/mK], all possibly varying in space (for nonhomogeneous solids like a chip with spreader) but not in time. As for convective exchanges, one can refer them to some "bulk" ambient temperature $T_a(t)$ with linear relationships like

$$\phi(x, y, z, t) = \gamma(T(x, y, z, t) - T_a(t)) \tag{3}$$

where ϕ is the heat flux [W/m^2] leaving the solid and (x, y, z) is a point on its surface, so that (3) provides a boundary condition for (1): on any finite area $S : \{f_s(x, y, z) = 0 \cap (x, y, z) \in \Xi\}$ of the solid boundary surface, Ξ being a convenient compact set in $\mathbb{R}^3$, the leaving power $Q_S(t)$ $[W]$ is

$$Q_S(t) = \int_S \phi(x, y, z, t)\, dS \tag{4}$$

where parameter γ [W/m^{2}K] is called the convective heat exchange coefficient. The models obtained in this setting are linear and with spatially uniform boundary conditions. Consequently, there are established methodologies to obtain code for their numerical solutions [27]. However, this fact only holds as long as γ is constant and the exchanges with the environment can be referred to a *single* temperature $T_a(t)$. Removing these hypotheses means abandoning the linear context.

B. Modeling of Next-Generation Heat Dissipation Systems

Next-generation heat dissipation systems include significantly more complex phenomena than heat conduction in a solid [6]–[8]. The following list presents the resulting modeling challenges.

1) Heat Transfer in Confined Fluids: When dealing, for example, with coolants in heat exchangers, linear correlations such as (3) easily prove inadequate. More complex, nonlinear ones are required like (in the case of fully developed turbulent motion) the Dittus–Bölter equation [28]

$$Nu = 0.023 Re^{0.8} Pr^{0.4} \tag{5}$$

where Nu, Re, and Pr are, respectively, the Nusselt, Reynolds, and Prandtl numbers. The Nusselt number contains γ implicitly. Similar relationships exist for other motion conditions. Another problem is that contrary to ambient air, coolants may sometimes undergo so significant thermal condition variations that require accounting for the dependence of their properties (most notably, density and specific heat) on temperature and possibly pressure [15]. The equations required to account for these phenomena introduce additional nonlinear terms and increase the model complexity significantly.

2) Heat Pumping Systems: This category comprises all systems where work (in any form) is spent to transfer heat in apparent contrast with the second principle of thermodynamics (e.g., TECs). Denoting the "cold" and "hot" side of the pump by a "c" and an "h" subscript, respectively, in MPSoC applications, one can most often use synthetic models such as

$$\begin{cases} Q_h(t) + Q_c(t) + W(t) = 0 \\ Q_c = \eta_{CC}(T_h(t), T_c(t))\eta(T_h(t), T_c(t), W(t)) \end{cases} \tag{6}$$

where Q_c and Q_h are the (signed) heat rates from the cold and hot sides of the pump, W is the power consumed by the pump itself,

$$\eta_{CC}(T_h(t), T_c(t)) = \frac{T_c(t)}{T_h(t) - T_c(t)} \tag{7}$$

is the Carnot cooling efficiency, and $\eta(T_h(t), T_c(t), W(t))$ is a characteristic function of the particular device, in the range (0, 1), which acts as an efficiency multiplier accounting for nonidealities. The previous models need to connect to the differential equations for energy balance in the fluid (e.g., air or water) to which heat is ultimately released, thus further increasing nonlinearity and overall complexity.

3) Fluid Circuits: When fluids move, hydraulics are relevant for two major reasons.

1) Besides energy balances, one has to take into consideration mass balance equations as well. These, in turn, require to relate flowrates with pressure drops, hence bringing into play other nonlinearities to compute the effect of viscous forces or "Darcy friction" [15].
2) If pressure drops are significant enough, these can provoke variations in the fluid properties, most notably density and specific heat. This means that hydraulic equations and thermal equations get coupled together, increasing complexity [29].

In addition, further nonlinearities appear when computing the powers needed for pumps and fans, which need to be accounted for in energy-related studies.

C. Mathematical Modeling Considerations

We shall now discuss, compatibly with space limits, how to turn the above models—made of equations—into algorithms for their solution. The key conclusion shall be that in this respect, the models addressed in Sections III-A and III-B call for radically different approaches.

1) Solids With Conventional Heat Dissipation: We start by reconsidering the assumptions used in Section III-A. First, to turn the partial differential (1) into an ordinary one in time, the most common approach is called finite volume. It consists of assuming that in a small enough volume, all time-varying quantities are spatially uniform. There are many ways to select the said small volumes to fill the entire solid to simulate. Here, we show in Fig. 1 the most common choice, adopted by 3D-ICE among other simulators, namely, parallelepipeds.

With reference to Fig. 1, consider, thus, a parallelepiped volume centered in $\{x_i, y_j, z_k\}$, where the finite sequences $\{x_i\}$, $i = 1, \ldots, N_x$, $\{y_j\}$, $j = 1, \ldots, N_y$, and $\{z_k\}$, $\ell = 1, \ldots, N_z$ are discretizations of the space occupied by the solid along the thee axes. Then, denote by $T(i, j, \ell, t)$ the volume temperature

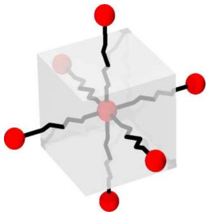

Fig. 1. Finite volume discretization of a solid, showing one volume and the thermal conductancs toward its six neighbor volumes.

at time t, by $C(i,j,\ell)$ its thermal capacity, by $P(i,j,\ell,t)$ the power generated in it, and by $G_{x,y,z}(i,j,\ell)$ its center-to-side thermal conductance along the axes. The energy balance for that volume shall read

$$\begin{aligned} C(i,j,\ell)&\frac{dT(i,j,\ell,t)}{dt}\\ &= P(i,j,\ell,t)\\ &\quad + G_x(i,j,\ell)(T(i-1,j,\ell,t)+T(i+1,j,\ell,t)-2T(i,j,\ell,t))\\ &\quad + G_y(i,j,\ell)(T(i,j-1,\ell,t)+T(i,j+1,\ell,t)-2T(i,j,\ell,t))\\ &\quad + G_z(i,j,\ell)(T(i,j,\ell-1,t)+T(i,j,\ell+1,t)-2T(i,j,\ell,t)). \end{aligned} \tag{8}$$

By expanding the products and grouping by the state variables (temperatures), this can be rewritten as

$$\begin{aligned} &\frac{dT(i,j,\ell,t)}{dt}\\ &\quad= \frac{1}{C(i,j,\ell)}P(i,j,\ell,t)\\ &\qquad - \frac{2\big(G_x(i,j,\ell)+G_y(i,j,\ell)+G_z(i,j,\ell)\big)}{C(i,j,\ell)}T(i,j,\ell,t)\\ &\qquad + \frac{G_x(i,j,\ell)}{C(i,j,\ell)}(T(i-1,j,\ell,t)+T(i+1,j,\ell,t))\\ &\qquad + \frac{G_y(i,j,\ell)}{C(i,j,\ell)}(T(i,j-1,\ell,t)+T(i,j+1,\ell,t))\\ &\qquad + \frac{G_z(i,j,\ell)}{C(i,j,\ell)}(T(i,j,\ell+1,t)+T(i,j,\ell+1,t)). \end{aligned} \tag{9}$$

For volumes facing the solid boundary, denoting by $T_a(t)$ the ambient temperature, terms in the form

$$\gamma S_{x,y,z}(i,j,\ell)\big(T_a(t)-T_{Sx,Sy,Sz}(i,j,\ell,t)\big)$$

shall be added to the right-hand side of (8)—hence of (9)—to represent convective exchange contributions, $S_{x,y,z}$ being the area(s) of the volume face(s) on the boundary, and $T_{Sx,Sy,Sz}$ being their *surface*—not center—temperature(s), computed based on the center one and the material conductivity. Note that we referred here to a simple geometry for brevity, but by collecting all the equations like (8) for all the volumes in any solid of interest, one ends up with a single system of ordinary differential equations. This system can be large, but has two important properties that can be exploited for an efficient solution.

1) First, as the C and G parameters do not vary with time, it is a system of *linear*, *constant coefficients* differential equations, and can, thus, be written as follows:

$$\frac{dT(t)}{dt} = AT(t) + BU(t), \quad U(t) = \begin{bmatrix} P(t) \\ T_a(t) \end{bmatrix} \tag{10}$$

where $T(t)$ and $P(t)$ are vectors containing all temperatures and powers, while A and B are constant matrices of convenient dimensions.

2) Second, as each volume exchanges only with its neighbours, matrix A is invariantly highly sparse.

Then, to solve (10) numerically a possible way, which is the one used by 3D-ICE, is to choose a timestep, h to name it, and to replace each time derivative with the incremental ratio over an h time span. At the generic step k—i.e., when $t = kh$—this means setting

$$\left.\frac{dT(t)}{dt}\right|_{t=kh} = \frac{T(kh)-T((k-1)h)}{h} \tag{11}$$

or, with an equivalent notation

$$\left.\frac{dT(t)}{dt}\right|_{kh} = \frac{T(k)-T(k-1)}{h}. \tag{12}$$

Substituting (12) into (10), one gets

$$\begin{aligned} \frac{T(k)-T(k-1)}{h} &= AT(k) + BU(k)\\ &\Rightarrow (I-hA)T(k) = T(k-1) + hBU(k) \end{aligned} \tag{13}$$

where I is the identity matrix.

The previous system is not differential anymore, and can be solved at each step to get $T(k)$. Even though the size of $I-hA$ scales with the square of the number of volumes, an efficient solution is possible using sparse solvers, as—recalling (9)—every row of $I-hA$ has at most seven nonzero elements.

The procedure just described broadly summarizes the current state of the art in MPSoC thermal simulators, if not for differences—inessential herein—in the heat dissipation models and the integration method. Most notably, (13) clearly shows the monolithic nature of the problem formulation; every volume, be it in the chip or the heat sink, ends up in the same set of equations, and the interconnection of volumes is completely self-similar.

From an implementation viewpoint, a thermal simulator has to construct data structures to perform the model integration starting from a 3-D geometry, then flatten this into the 1-D T vector, and finally, construct the above matrices consistently. This is an inherently error-prone task that requires careful testing and validation of the entire codebase at every change, to avoid the introduction of subtle bugs.

Summing up, as long as geometries are "standard" enough so that the task above can be carried out once and for all, the monolithic approach can be very efficient. But, as for the extensibility of a thermal model beyond such a context, the same approach is a very critical barrier that we propose to overcome in 3D-ICE 3.0.

2) Modern Heat Dissipation Systems: Based on the previous analysis, we propose to reevaluate Section III-B. To get to the conclusions as straightforwardly as possible, we slightly rephrase two findings from the previous section.

1) Each volume of solid exchanges heat only with its six neighbors (solid or air).
2) The said heat exchanges depend linearly on the difference of the neighboring temperatures.

For this section, neither of these two conclusions hold. Consider, for example, a fluid stream (such as coolant between two fins) exchanging heat with a containment boundary (e.g., locally, the fins themselves). Then, let the fluid thermodynamic coordinates be the pressure p and the enthalpy h, as is common practice in fluid modeling [30]. To describe such a component (in the absence of flow reversals for practical reasons), we need the following.

1) The fluid mass balance (including the momentum equation in the form of pressure/flowrate relationships)

$$\begin{cases} M_f(t) = V\rho(p(t), h(t)) \\ \frac{dM_f(t)}{dt} = w_i(t) - w_o(t) \\ w_i(t) = f_f(p_i(t), p(t), k_f(p(t), h(t), w_i(t))) \\ w_o(t) = f_f(p(t), p_o(t), k_f(p(t), h(t), w_o(t))) \end{cases} \tag{14}$$

where M is the contained mass, V is the element volume, ρ is the fluid density, $w_{i,o}$ are the inlet and outlet mass flowrates, $p_{i,o}$ are the pressures of the preceding and following stream element in the direction of flow, $f_f(\cdot,\cdot,\cdot)$ is a convenient flow correlation, and k_f is the friction factor that, in turn, depends on the fluid thermodynamic condition and (through the Reynolds number) velocity, i.e., on flowrate (we neglect gravity effects for simplicity). Note that (14) is inherently nonlinear and implicit [15]. It can be simplified by assuming an incompressible fluid, but the above two characters in the flow equation(s) remain.

2) The fluid energy balance results as follows:

$$\begin{cases} E_f(t) = M(t)e(p(t), h(t)) \\ \frac{dE_f(t)}{dt} = w_i(t)h_i(t) - w_o(t)h(t) + Q_{cf}(t) \\ Q_{cf}(t) = \gamma(t)S_c(T(p(t), h(t)) - T_c(t)) \\ \gamma(t) = f_\gamma(p(t), h(t), w_o(t))) \end{cases} \tag{15}$$

where E is the contained energy, e is the specific internal energy, h_i is the enthalpy in the preceding element in the direction of flow, Q_{cf} is the heat rate received from the containment wall, S_c is the surface of that wall, T_c is the temperature of that surface, and γ is the heat transfer coefficient depending (e.g., via the Dittus–Bölter law) on its thermodynamic state and velocity, i.e., flowrate. Also (15) is inherently nonlinear and implicit.

3) The containment energy balance is

$$\begin{cases} E_c(t) = M_c c_c T_c(t) \\ \frac{dE_c(t)}{dt} = -Q_{cf}(t) + Q_{ec}(t) \end{cases} \tag{16}$$

where M_c is the containment element mass, c_c is its specific heat, T_c is its temperature (assumed uniform in the radial direction for simplicity), and Q_{ec} is the heat rate from the external environment to containment. Note that in the envisaged case of a finned heat sink, the above would need completing with a description of the temperature field in the whole finned plate. This system is frequently too complex to be modeled with a standard parallelepiped-based approach [15], [29].

In addition, the geometries of modern heat dissipation systems are extremely variable, as it occurs in case of the liquid piping inside a rack, or in the case of fluid paths within plate exchangers, of heat pumps, TECs, impellers, storages, and the like. The interconnection of elements described, e.g., by (14) through (16), is absolutely not keen to be described with matrices simply. Finally, even if some way of the standardizing geometrical connections could be found, nonlinear equations do not combine together smoothly as linear ones naturally do. A small modification in just the interconnection of components described by *implicit* systems like the above can alter the characteristics of the compound system of equations dramatically, with a very strong impact on the solution procedure.

Summing up, the examples we reported in this section should convince the reader that the linear finite volume abstraction is straightforward and efficient for simple solids, but completely inadequate for the next generation of MPSoC systems with complex cooling and heat sink technologies.

As a consequence, we propose to describe such systems with an equation-based modeling approach, owing to its potential and flexibility in tackling complex differential and algebraic models [31]. In particular, we choose the Modelica language, because with that language the modeling, for example, of heat pipes [32], TECs [33], liquid cooling elements [34], thermo-hydraulic systems [35]—also encompassing high-precision substance property calculations [36]—is already deeply studied, and model libraries to build upon are available. This is why, in this article, we do not expand further on how to solve the equations in this section, but rather concentrate on providing an interface between 3D-ICE and the equation-based modeling world. No doubt some research effort shall be devoted also to create equation-based models specifically targeted to the addressed domain, but this will be treated in future works.

D. Supporting Thermal Co-Simulation

To overcome the limitations of a monolithic linear thermal simulator, we propose a co-simulation [37]–[39] approach that partitions the thermal simulator in two cooperating parts. One is devoted to the chip and is based on an optimized implementation taking advantage of the linear nature of the problem to maximize performance. The latter can use nonlinear solvers and automatic code generation [40], [41] from an equation-based *modeling*—not programming—language to handle the variability of next-generation heat dissipation solutions.

By standardizing the interface between the two solvers, an extensible solution can be achieved, where even completely different heat sink models can be *plugged* into an unmodified chip model. As the chip model source code is not modified, there is no possibility of introducing bugs, and the associated validation burden is removed.

In the following paragraph, we discuss how the differential equations of a monolithic thermal model can be partitioned while at the same time introducing nonlinear simulation capability. Starting from the monolithic model of (10), the partition operation translates in splitting the state vector and expanding the input vector as follows:

$$T(t) = \begin{bmatrix} T_c(t) \\ T_s(t) \end{bmatrix}, U_c(t) = \begin{bmatrix} P(t) \\ -P_i(t) \\ T_a(t) \end{bmatrix}, U_s(t) = \begin{bmatrix} P_s(t) \\ P_i(t) \\ T_a(t). \end{bmatrix} \quad (17)$$

The split of $T(t)$ defines $T_c(t)$, the state vector of the differential equation model for the chip, and $T_s(t)$, the one for the sink. The input power vectors need instead to be redefined to consider the heat exchanged across the interface between the two models. $U_c(t)$ is the input power vector, being composed by $P(t)$, the power generated inside the chip, $P_i(t)$, the power exchanged with the heat sink (the minus comes from the passive sign convention for the co-simulation interface), and the ambient temperature $T_a(t)$. $U_s(t)$ is the input power vector for the heat sink. Other than $P_i(t)$ and $T_a(t)$, here we find $P_s(t)$, which is the power generated within the heat sink. This term is usually zero unless in the case of TECs or other cooling solutions that dissipate additional power.

Having defined the new state and input vectors, it is now possible to separately express the chip and heat sink differential equation models. Starting from the chip model, we can write it as follows:

$$\begin{cases} \frac{\mathrm{d}T_c(t)}{\mathrm{d}t} = A_c T_c(t) + B_c U_c(t) \\ T_{ci}(t) = X_c T_c(t) \end{cases} \quad (18)$$

where A_c represents the interconnections between the chip volumes, and B_c is constructed from B considering that the interaction between the volumes of the chip and heat sink has now become boundary conditions for the chip model.

The new vector T_{ci} contains the chip volume temperatures at the interface with the sink, and is obtained by means of the selection matrix X_c, made of ones and zeros. This vector is an output of the first model, passed to the heat sink thermal model at every time step. The heat sink thermal model is instead nonlinear and reads

$$\begin{cases} F\left(\frac{\mathrm{d}T_s(t)}{\mathrm{d}t}, T_s(t), U_s(t), \theta_s\right) = 0 \\ T_{si}(t) = X_s T_s(t) \\ P_i(t) = G(T_{ci}(t) - T_{si}(t)) \end{cases} \quad (19)$$

where the first equation is the most general form of a nonlinear implicit differential equation system with parameters θ_s. The second equation computes the temperatures at the sink cells in contact with the chip. The last one computes the heat exchanged between chip and sink $P_i(t)$, G being the matrix of thermal conductance values at the interface. $P_i(t)$ is an output of the heat sink co-simulation interface, passed back to the chip model at every time step.

The two resulting systems of differential equations can then be independently discretized and numerically solved, for example, using the method described earlier.

IV. 3D-ICE 3.0 Co-Simulation Architecture Design

The co-simulation approach just described has been implemented to conceive the new 3D-ICE 3.0 thermal simulator. Fig. 2 shows its software architecture, detailing which components have been extended compared to the baseline 3D-ICE, and which are entirely new.

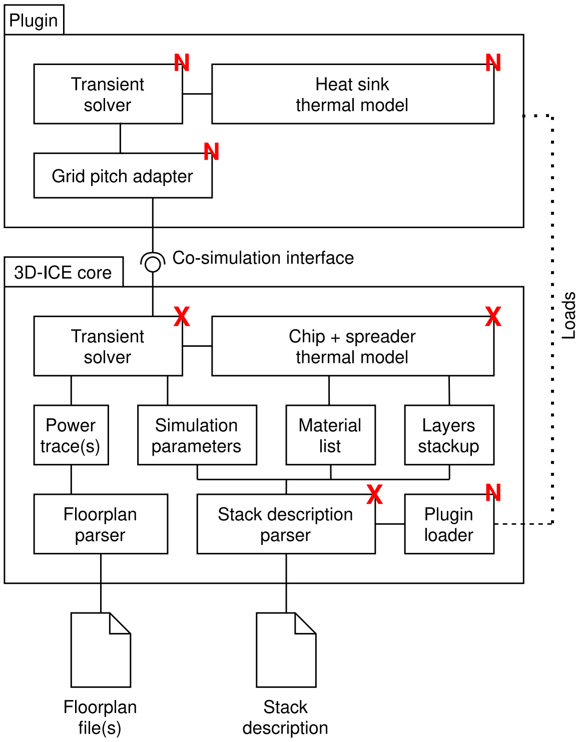

Fig. 2. Architecture of 3D-ICE 3.0. New components compared with previous 3D-ICE versions are marked as N, while extended ones with X.

In the chosen co-simulation infrastructure, the 3D-ICE core simulates the parts of the system that require only configuration and can be modeled using linear differential equations (i.e., MPSoC and heat spreader), while the plugin is dedicated to the parts that are nonlinear and need replacing to achieve the required level of flexibility (i.e., heat sink).

In addition, an important part of the 3D-ICE 3.0 thermal model is the heat spreader component, as it acts as the interface between the core and plugin. The temperatures and heat flows that are exchanged with the plugin as part of the co-simulation process are those of the heat spreader. In case the MPSoC to be simulated lacks a heat spreader, the last material layer that is physically connected to the heat sink takes its place, such as the thermal interface material or solder. The plugins are implemented as shared libraries, and a loader component has been added to load the desired plugin at runtime based on the stack description configuration.

A. Co-Simulation Interface

Our novel 3D-ICE 3.0 co-simulation interface follows the FMI [16], [42], [43] standard, an industry standard for providing interoperability among models based on differential equations. In particular, FMI models are organized using a common structure, including an XML file describing the variables and parameters that are exposed across the interface. All FMI models provide the same set of *C* functions, and the lifecycle of a model instance is standardized. FMI allows to raise the abstraction level at which a model can be built as models need not be written directly in *C*. The development environments of a number of high-level modeling languages, such

TABLE II
IMPORTANT DIFFERENCES BETWEEN ALGORITHMIC AND EQUATION-BASED MODELING

Action	Monolithic algorithmic approach	Modular equation-based approach
Model a supported heat sink configuration (e.g., change fin dimension)	Edit configuration file	Edit the model parameters
Model an unsupported heat sink configuration (e.g., change liquid cooling circuit layout)	Extend and re-validate the core	Assemble existing components
Model a heat sink with unsupported physical phenomena (e.g., introduce evaporative cooling)	Extend and re-validate the core	Create and connect the required new components
Validation of new components individually	Not possible	Possible
Validating the model of a new configuration	Impacts the entire core	Not required as long as the used components are individually validated

as Modelica and Simulink/Simscape, allow to automatically produce an FMI from an high-level model, which can then be used with 3D-ICE 3.0 to simplify greatly the exploration effort for computing system designers.

Among the languages supporting FMI, Modelica [44] is a domain specific language for differential equations modeling. Its main advantage is that it allows to write differential equations directly, as opposed to writing the code to solve them. A Modelica compiler uses symbolic manipulation to transform the equations in the *C* code to solve them. Thus, it automates the complex and error-prone task of manually writing optimized code to solve differential equations. As a result, 3D-ICE 3.0 comes with a Modelica library providing basic heat sink building blocks, such as base plates and fins, as well as a base class that can be extended to implement the co-simulation interface with 3D-ICE. Indeed, the two heat sink models we describe in Section V illustrate this by making use of the provided Modelica library.

B. Grid Pitch Adaptation

When simulating a complex MPSoC, a high level of detail is often desired or even required in order to observe relevant phenomena such as hot spots. However, spatial discretization of the device to be simulated using a fine mesh increases the size of the problem significantly and makes the model solution computationally intensive [26]. It should be considered that a heat sink is physically many times larger than an MPSoC, and its features are orders of magnitude larger. Hence, a heat sink model discretized with the same level of detail of the chip requires a considerably larger number of volumes and equations, without bringing benefits in terms of simulation accuracy and detail. This effect is further amplified as nonlinear equations are involved in the heat sink model, due to the greater computational complexity of nonlinear solver algorithms compared to the linear case.

To overcome this issue, 3D-ICE 3.0 includes a grid pitch adaptation component that allows to perform simulations with two different discretization grids across the co-simulation boundary. Therefore, by simulating only the MPSoC using a fine mesh, and using a coarser mesh for the complex nonlinear equations of the heat sink, performance can be substantially improved, as will be shown in Section VII.

C. Paradigm Shift in Heat Sink Modeling

We conclude the presentation of the 3D-ICE 3.0 architecture by evidencing the practical advantages of the newly introduced equation-based modeling capability.

With an algorithmic modeling approach, one does not really write the model—which is a set of differential and algebraic equations—but rather its solution algorithm. The model to algorithm translation is, therefore, a task of the developer. This translation is complex, time consuming, and error prone, and requires competence in numerical integration. Moreover, since changing only the form of an equation in a system of equations can change the aspect of the solution completely, whenever a new model differs from an existing one by anything else than the numeric value of some parameters, the above translation needs to be reevaluated and possibly redone from scratch.

However, allowing the user to just edit parameters in a configuration file suffices to address only a limited variety of physics, and the consequences from the viewpoint of heat sink modeling are outlined in Table II. Summarizing, we could say that the proposed modular equation-based approach separates the role of component modeling and system modeling. An expert in physics can create models of new components (e.g., a liquid pipe) and validate them individually, e.g., with extensive laboratory tests. A computer architecture scientist can take those components, assemble them together (e.g., creating a liquid cooling circuit), set component-level parameters (e.g., the pipe length and diameter), and rest assured that the compound model of the circuit will be correct and consistent. Decades of experience in other domains, such as process control and vehicles, support this modeling strategy, whence our proposal to make thermal simulators capable of exploiting it.

D. Example of Heat Sink Modeling in 3D-ICE Using Modelica

To exemplify how differential equation models appear in Modelica, we describe here a model with a square copper plate of 4-cm side and 1-mm thickness used as heat sink toward ambient air. The Modelica code in Listing 1 models the plate as a $2 \times 2 \times 1$ finite volume grid. Although the spatial discretization is hardcoded for simplicity, it can be made parametric. This model uses the Modelica object-oriented features, extending (the Modelica word for inheriting) the `Heatsink` base class from the 3D-ICE heat sink library. This base class provides the `initialTemperature` variable as well as the `bottom` heat port, which has to be connected to the center cells of the bottom-most layer of volumes of the heat sink. The base class also requires certain parameters to be set, as shown in lines 3–5. Note that the order of declaration is irrelevant in Modelica, so line 3 can refer to line 7. Lines 7–21 declare physical parameters, and lines 22–26 are

```
1  model NonlinearHeatsink
2    extends HeatsinkBlocks.PartialModels.Heatsink(
3      bottomLength = L, bottomWidth  = L,
4      bottomRows   = 2, bottomCols   = 2,
5      cellBottomConductance = gz);
6    //Copper plate parameters
7    parameter Length L = 0.04  "Plate length [m]";
8    parameter Length t = 0.001 "Plate thickness [m]";
9    parameter Density rho             = 8960  "[kg/m^3]";
10   parameter SpecificHeatCapacity cp = 384.6 "[J/(kg.K)]";
11   parameter ThermalConductivity k   = 401   "[W/(m.K)]";
12   parameter HeatCapacity c = cp*rho*(L/2)^2*t;
13   parameter ThermalConductance gz  = (L/2)^2/(t/2)*k;
14   parameter ThermalConductance gxy = (L/2)*t/(L/4)*k;
15   //Air parameters
16   parameter Temperature airTemp = initialTemperature;
17   parameter Velocity velocity = 5 "[m/s]";
18   parameter Density rhoAir = 1.1;
19   parameter SpecificHeatCapacity cpAir = 1020;
20   parameter ThermalConductivity kAir = 0.016;
21   parameter DynamicViscosity muAir = 1.182e-5;
22   CoefficientOfHeatTransfer gamma;
23   Real Re, Nu, Pr; //Reynolds, Nusselt, Prandtl numbers
24   Temperature T[2,2](each start = initialTemperature);
25   Temperature Ttop[2,2]; // Top plate temperature
26   Power Qair[2,2]; //Heat exchange of top plate with air
27 equation
28   c*der(T[1,1]) = (T[1,2] + T[2,1] - 2*T[1,1])*gxy
29                 + bottom[1,1].Q_flow + Qair[1,1];
30   c*der(T[1,2]) = (T[1,1] + T[2,2] - 2*T[1,2])*gxy
31                 + bottom[1,2].Q_flow + Qair[1,2];
32   c*der(T[2,1]) = (T[1,1] + T[2,2] - 2*T[2,1])*gxy
33                 + bottom[2,1].Q_flow + Qair[2,1];
34   c*der(T[2,2]) = (T[1,2] + T[2,1] - 2*T[2,2])*gxy
35                 + bottom[2,2].Q_flow + Qair[2,2];
36   Re = rhoAir*velocity*L/muAir; //L=characteristic length
37   Nu = gamma*L/kAir;
38   Pr = muAir*cpAir/kAir;
39   Nu = 0.023*Re^0.8*Pr^0.4; //Dittus-Boelter
40   for i in 1:2 loop for j in 1:2 loop
41     bottom[i,j].T = T[i,j];
42     Qair[i,j] = (Ttop[i,j] - T[i,j])*gz;
43     Qair[i,j] = gamma*(L/2)^2*(airTemp - Ttop[i,j]);
44   end for; end for;
45 end NonlinearHeatsink;
```

Listing 1. Modelica heat sink example.

the model variables. Line 24 declares the state variables (the volume center temperatures). The rest of the model contains the equations that describe the physical phenomena. Note the use of the `der()` operator to express derivatives in differential equations. Note also that all those lines are equations and not assignments, so for example lines 42 and 43 are not two consecutive assignments, but rather two equality constraints to always hold true. Finally, the Dittus–Bölter equation is written in its native implicit form (line 39); the Modelica tool solves for the exchange coefficient by automatic manipulation. The code shown is provided as part of the 3D-ICE 3.0 release.

V. New Validated Heat Sink Models in 3D-ICE 3.0

3D-ICE 3.0 includes two new heat sink models, namely, an air cooled heat sink including a fan model to simulate forced convection and a water heat sink used in liquid cooling setups. Both are commercial off-the-shelf components. Fig. 3 shows the two heat sinks connected to the thermal test chip (TTC) platform. Model parameter fitting and validation was done using experiments performed with a TTC platform, as explained in Section VI.

A. Air Heat Sink Plus Variable Speed Fan Model

The air heat sink plus fan is a model of an HS483-ND copper heat sink coupled with a P14752-ND fan from Digi-Key.

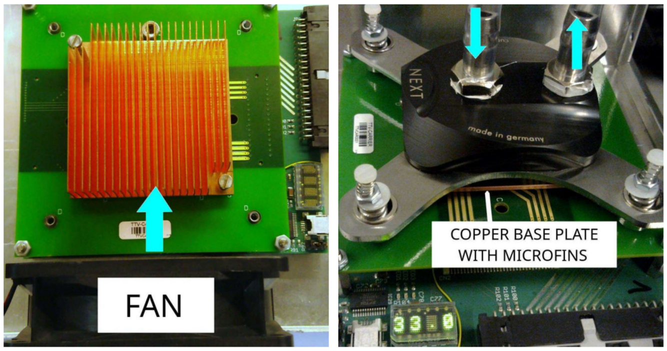

Fig. 3. Air heat sink with fan (left image) and water heat sink (right image) installed on the TTC to collect experimental data.

The model supports both natural and forced convection, and can simulate the effect of fan speed variations as well as the possibility to turn off and on the fan throughout the simulation. Thus, it can switch between natural and forced convection on the fly. It should be noted that, as shown in Fig. 3, the model assumes that the heat sink is placed horizontally, and the fan is placed at the side of the sink, blowing air through the fins, mimicking the setup of enterprise servers. Thus, as the fan does not obstruct the air flow upward from the sink, the same natural convection model can be used both to simulate a pure natural convection set up, where the fan is not present, as well as a forced convection setup where the fan is turned off.

1) Fin and Base Model: To model the heat sink base plate, a uniform grid of finite volumes is used. The bottom of this grid is connected to the heat spreader across the co-simulation interface. The 3D-ICE 3.0 grid pitch mapping component takes care of adapting the grid of the base plate to the one of the underlying chip and heat spreader, as well as of connecting only the relevant cells of the base plate in the common case of a heat spreader smaller than the sink base.

Also, the heat sink fins are modeled as finite volumes, and they exchange heat on both sides with air. The heat transfer coefficient is constant for the natural convection case, while it is a nonlinear function of the air velocity in the forced convection case. The heat sink data sheet provided characteristic curves of the compound thermal resistance as a function of the air velocity, from which the heat transfer coefficient was derived. The obtained points were fitted with a quadratic function. The laminar air flow was assumed and verified through Reynolds number computation.

2) Fan Model: A fan at a constant speed produces an air flow that depends on its characteristics and those of the airflow resistance that the air travels. When both the fan inlet and outlet are completely unobstructed, a fan produces the maximum volumetric air flow rate $Q_{\max}$ and a zero-pressure differential, while with a completely obstructed path a fan produces the maximum pressure differential at a zero air flow.

When connected to a heat sink, the fan works at an operating point in between the aforementioned extremes, depending on the sink airflow resistance. Moreover, from the air flow, it is possible to compute the air speed along the heat sink fins, which is needed to compute the heat transfer coefficient.

The fan model used is a performance optimized, approximate uniform airflow model that takes as input the fan speed in RPM and produces the air speed. The relation between the fan speed and $Q_{\max}$ has been taken from the fan data sheet, a reference area of 115% of the fan cross section was assumed to compute the air velocity, as the fan is not ducted. In the impossibility of taking spatially distributed pressure measurements, a single free parameter was assumed as the per unit flow reduction caused by the fins hydraulic resistance. This operating point parameter was fitted using transient thermal experiments to reproduce the data.

B. Water Heat Sink Model

The water heat sink is a cuplex kryos NEXT 21606 water block designed for Intel CPUs. It is composed of a copper base plate, with a microfin array at the center. Atop that plate sits an acetal plastic dome that together with a gasket forms a water tight seal (see Fig. 3). Water is injected though a slit in the dome and flows from the center of the fins toward both ends, exiting through an asymmetrical path as one of the fin ends is closer to the outlet than the other. The base is modeled using a uniform grid of finite volumes just like the air heat sink. Due to the number of microfins and the fine pitch between them, modeling them individually would have required a high number of equations and a considerable simulation performance penalty without providing a significant benefit in terms of modeling accuracy. To optimize the model, we thus decided to simulate fins in groups of four instead of individually, trading some accuracy for simulation speed.

The data sheet of this heat sink did not provide any characteristic curve nor data about the thermal resistance. For this reason, the model had to be fitted entirely from transient thermal experiments. Based on geometrical dimension measurements, the cross sectional area of the headroom between the fins and the dome was obtained, which made it possible to compute the water speed as a function of the flow rate. As the water path is asymmetrical, the model takes into account the corresponding nonuniform water flow at the two fin halves, which had to be inferred from the different chip temperatures at the opposite sides of the slit. As with the air heat sink, to support variable flow rates, a correlation of the heat transfer coefficient as a function of the water speed is required. In the absence of manufacturer data, also this correlation had to be measured from transient thermal experiments. In this case, a cubic fitting was chosen as it proved to reduce the error significantly compared to a quadratic one.

VI. Experimental Validation

We now present a complete set of experiments where we apply a known power profile to a real chip connected to an air and a water heat sink. Then, we record temperature traces at known locations in silicon, and finally, we compare them to those produced by the corresponding chip plus sink model. This procedure allows us to both validate our models, and demonstrate the relevance of good sink modeling for an accurate representation of chip temperatures.

A problem to face with such experiments is that in MPSoCs one can measure the total power consumption, but not precisely its spatial distribution [45]. Furthermore, the exact location of temperature sensors is often proprietary information, hence also mapping temperature readings to precise points in silicon is difficult.

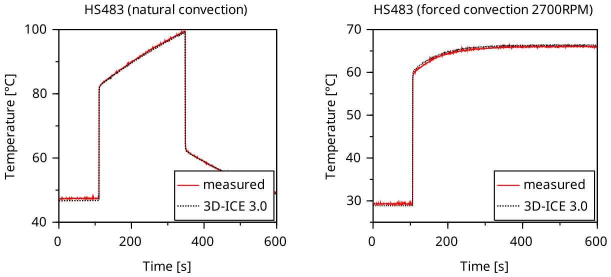

Fig. 4. Comparison of the 3D-ICE temperature trace (black, dotted) with that from the TTC (red, solid) for the air heat sink caused by a 200 W/cm^2 hot spot. Temperature shown is that of the temperature sensor closest to the hot spot. Natural convection (left) versus forced air convection (right).

To overcome the aforementioned issues, we performed the validation using a dedicated platform [46] based on a TTC, i.e., a special integrated circuit consisting of an array of heater elements and temperature sensors. The quoted platform completes the TTC with the required electronics and firmware to apply arbitrary power patterns and measure the corresponding temperatures. We created a model of the TTC using two layers of finite elements cells to account for the active silicon and bulk. Thus, we modeled the TTC floorplan using manufacturer data as for the heater element sizes and locations, and the temperature sensor positions. Both the air heat sink (in the natural and forced convection mode of operation) and the water heat sink described in Section V are considered in this validation.

The model of the air heat sink was validated using traces from an open data set [47] with the same TTC platform. The data set includes 24 traces, where half were collected with the heat sink operating under natural convection, and half under forced air convection. All traces are about 1-h long and consist of a preheat operation followed by a thermal transient with different amplitude and spatial distribution.

Table III shows the average and maximum error of the air sink model across all the 24 traces in the data set for the natural and forced convection case. As this table shows, the average error is less than 1 °C, and it is higher in the natural convection case due to simplifying assumptions in the air model introduced to improve simulation speed, a matter that is more significant with increasing air temperature differentials.

Then, Fig. 4 shows a sample of the output of our proposed 3D-ICE 3.0 thermal co-simulation compared against two traces for a 200-W/cm^2 hot spot case. The left figure shows the hot spot temperature when the heat sink is operating under natural convection, while the right one under forced air convection with the fan operating at 2700RPM. In both cases, the hot spot is activated after a 5-W/cm^2 uniform power preheat phase, hence the different initial temperatures. In the natural convection case, a thermal protection occurred during the experiment as the TTC temperature reached 100 °C. In this case, the TTC control circuit cuts the chip power, resulting in a second transient. The maximum error in this experiment is really low, namely, 2.4 °C for the natural convection case and 1.4 °C for the forced convection.

TABLE III
VALIDATION OF THE 3D-ICE CORE AND CO-SIMULATION INTERFACE CONNECTED TO THE HEAT SINK MODELS

Error	Air heat sink, natural convec.	Air heat sink, forced convec.	Water heat sink
Average	$0.8°C$	$0.5°C$	$0.9°C$
Maximum	$4.1°C$	$3.0°C$	$5.0°C$

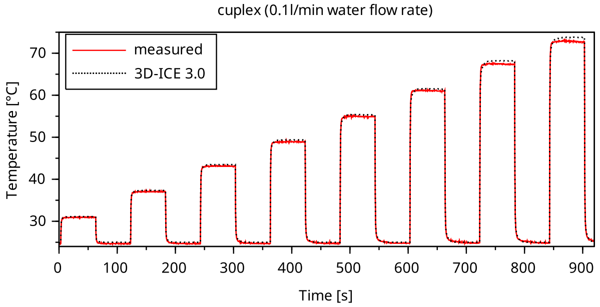

Fig. 5. Comparison of the 3D-ICE temperature trace (black, dotted) with the measured trace from the TTC (red, solid) for the water heat sink. Uniform heating in eight power steps starting at 10 W and ending at 80 W.

Next, the validation of the water heat sink was performed similarly, by connecting it to the TTC platform and performing experiments with different power, power spatial distribution, and water flow rate. To accurately prescribe the water inlet temperature and flow rate, a LAUDA-Brinkmann water chiller was used. The average and maximum error across the entire experimental campaign is reported in Table III. The average error is again less than 1 °C, while the maximum error is higher than the air heat sink due to difficulty in modeling the nonuniform water flow across the fins caused by the heat sink geometry.

In Fig. 5, we present a sample from the experimental campaign consisting in applying progressively higher power steps, ranging from 10 to 80 W uniformly spread across the chip surface with a 0.1-l/min water flow rate. The maximum error in this experiment is really negligible (only 1.3 °C).

VII. Performance Assessment

An MPSoC thermal simulator is often required to produce detailed thermal maps for thermal-aware microarchitectural design, optimization, and exploration down to the functional unit level [12], [13]. Accurate simulation is, therefore, mandatory and speed desirable. In other cases, such as in the thermal policy design when a hardware in the loop is involved or in predictive thermal policies [48] where a simulator needs to operate in lockstep with its physical counterpart, an MPSoC thermal simulator may be conversely required to provide coarse-grain thermal results, but in real time. In these cases, speed is the necessity, and accuracy is the desire.

In this section, we first show that 3D-ICE 3.0 provides adequate performance to serve both needs, and in addition, that its grid pitch adaptation capability allows for a straightforward tuning toward either of the two.

To this end, we performed experiments using a laptop with a Core-i7-8750H CPU and 16 GB of RAM, running Ubuntu Linux version 18.04.4. In this case, 3D-ICE was compiled with GCC 7.5.0, and the heat sink models were translated with OpenModelica 1.16.0~dev-582-ga906312. The experiment consisted in the simulation of a 60-s power trace to obtain the corresponding temperature trace with both heat sinks (the air sink being used in forced convection mode, thus with also the fan being modeled). The experiment was repeated with a progressively greater level of detail in the chip discretization. In addition, the experiment was repeated without the grid pitch adaptation feature described in Section IV, thus with also an increasing heat sink level of detail. As with all thermal simulators, the simulation time largely depends on the number of equations of the model. In the case of 3D-ICE 3.0, it is additionally straightforward to separate exactly the time contribution of the chip (18), and those of the heat sink (19).

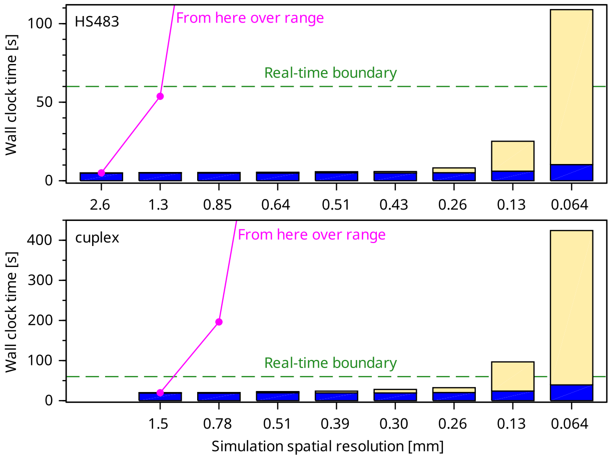

Fig. 6. Simulation time comparison with the air (top) and water (bottom) heat sinks as a function of the chip spatial resolution (bars), or the chip and heat sink spatial resolution (solid magenta line). Bars show the breakdown between the time spent in the 3D-ICE core (pale yellow) and heat sink model (dark blue).

The results are shown in Fig. 6. The bars represent the wall clock time required to perform the simulation, showing the breakdown between the time spent in the 3D-ICE core simulating the chip (pale yellow), and in the heat sink model (dark blue). The solid magenta line instead shows the total (3D-ICE core and sink model) simulation time without grid pitch adaptation. As this figure highlights, when the chip simulation detail is coarse, the majority of the simulation time is spent in the heat sink simulation. This fact is understandable as, due to the larger physical size, the heat sink model requires a significantly larger number of equations. For example, the chip model at 2.6-mm spatial resolution is composed of only 48 equations, while the air heat sink is composed of 2164 equations. The same is true for the water heat sink, which requires 3458 equations to be simulated at 1.5-mm resolution, while the chip only requires 147. As the chip resolution is increased, the correspondingly higher number of equations required causes the simulation times to increase, but thanks to our included grid pitch adaptation feature, the heat sink is always simulated at a level of detail determined by its geometry, not the chip geometries. Thus, most of the simulation time increase occurs only in the 3D-ICE core.

Then, experiments without grid pitch adaptation show the importance of this feature. Considering the air heat sink, increasing the heat sink resolution from 2.6 to 1.3 mm brings the heat sink number of equations from 2164 to 6396, and the total simulation time from 5.01 to 53.7 s, while increasing the resolution further to 0.85 mm (12 744 equations) increases the simulation time to 293 s. Considering that increasing the heat sink resolution from 2.6 to 0.85 mm is not justified due to its large geometries and that it only affords a reduction in the maximum error of 0.2 °C at a 58× performance penalty, it is evident that the grid pitch adaptation is a fundamental feature to achieve good simulation performance. Similar considerations apply to the water heat sink, where reducing resolution from 1.5 to 0.78 mm brings the total simulation time from 19.75 to 196.4 s, and a further resolution increase to 0.51 mm increases it further to 1040 s, a 53× performance penalty resulting in a reduction in the maximum error of only 0.6 °C.

A final experiment was performed to quantify the overhead of the co-simulation interface compared to a purely monolithic approach. In this experiment, we have reimplemented in Modelica the same heat sink model that was provided in previous versions of 3D-ICE (“top heat sink” configuration parameter in the 3D-ICE stack description file). We have performed several comparative experiments with different chip spatial discretization granularity ranging from 2500 to 40 000 equations, and the overhead of the co-simulation and grid pitch mapper was shown to be always below 10%. It should, however, be noted that in the general case the overhead of co-simulation can vary widely, depending on the type of differential equations that need to be solved at the two sides of the co-simulation boundary, their properties, and the computational complexity required for their solution.

In summary, when coarse-grain simulations are performed, real-time operation is achievable, while for detailed chip simulations, thanks to the possibility to simulate the chip and heat sink at different levels of detail, simulation times are dominated by the chip part of the simulation, and the possibility to simulate advanced cooling solutions does not significantly slow down thermal simulations compared to a more limited conventional thermal simulator.

VIII. Case Study

In this section, we analyze and compare the cooling performance of the air and water heat sinks by means of 3D-ICE 3.0 simulations. In all the shown experiments, the ambient temperature is set to 25 °C and the floorplan used is the same TTC one of the validation in the previous section.

The first experiment shows the capability of 3D-ICE 3.0 to provide high-resolution temperature maps. In particular, Fig. 7 shows the temperature map resulting from a nonuniform heating pattern, with one quadrant of the chip heated at 80 W/cm^2 and the rest at 5 W/cm^2. In this experiment, water cooling afforded a 12.2 °C temperature reduction in the hottest spot, and a 11.5 °C average temperature reduction.

Then, Fig. 8 shows instead the effect of a hot spot in a transient simulation, as well as the effect of variable coolant flow rate. In this experiment, one heater of the simulated TTC was set to 300 W/cm^2 at the start of the simulation, and this value is kept at constant power. Our results confirm that due to the fast thermal dynamics in integrated circuits [49], the initial temperature increase is very rapid and shows little dependence on the type of heat sink attached. In this experiment, temperature increases from 25 °C to 60 °C in just 92 ms with the air heat sink, and 42 ms with the water one. Then, the maximum temperature reached as well as the rate of temperature increase depend greatly on the thermal dynamics of the heat sink. More precisely, with the air heat sink operated under natural convection, temperature reaches 100 °C in 250 s, after which the fan is turned on to 1500RPM before the thermal transient is over. If the fan would not be turned on, the temperature would have reached 159.3 °C. The increased heat transfer reduces temperature to 81 °C in 400 s. Then, at 800 s, from the start of

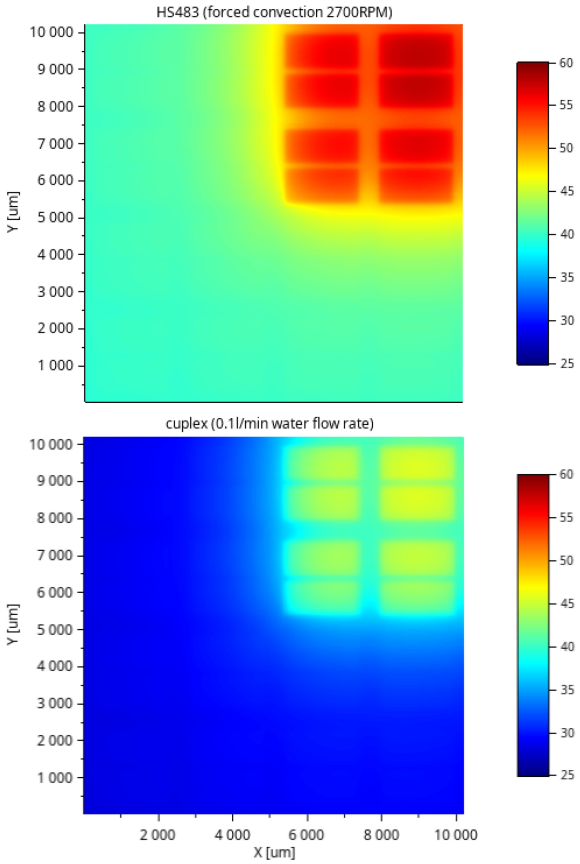

Fig. 7. 3D-ICE simulation comparing the air and water heat sinks. Four TTC heating elements heated at 80 W/cm^2 (top right), remaining elements heated at 5 W/cm^2. The shape of the TTC heating elements is visible.

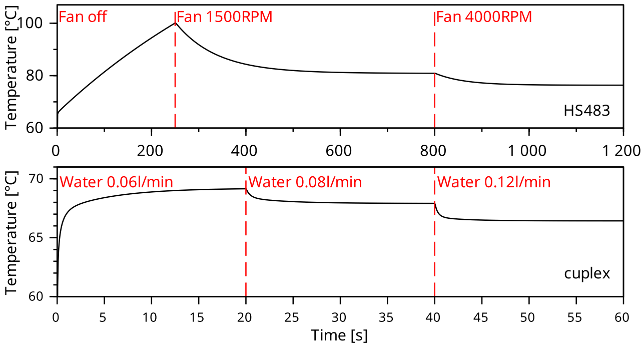

Fig. 8. 3D-ICE simulation showing the effect of variable coolant flow with a 300 W/cm^2 hot spot. The temperature shown here is the one at the hot spot.

the simulation, the fan speed is again increased to 4000RPM, bringing down the temperature to 76.4 °C. Furthermore, in accordance to theoretical calculations, increasing the fan speed not only decreases the temperature, but increases the speed of the thermal transient. The second temperature reduction is in fact complete in just 300 s.

The same hot spot when cooled using the water heat sink only reaches 69.2 °C with a water flow rate of 0.06 l/min, with the transient being complete after just 20 s. When the water flow rate is increased to 0.08 l/min, temperature drops to 67.9 °C in 15 s, while a further increase to 0.12/min reduces temperature to 66.4 °C in 12 s.

Overall, the study shows the advantages of water cooling but also evidences an increase in the speed of thermal transients, as previous publications have highlighted [7], [12], motivating the need for a newer thermal simulator to explore advanced cooling technologies. Moreover, a dependence of said transient times on the coolant flow rate has been observed, a matter that may affect thermal policies and whose effect—at least to the best of our knowledge—has not been accounted for.

IX. Conclusion

3D-ICE 3.0 is an MPSoC thermal simulator designed to simulate integrated circuits connected to next-generation heat dissipation solutions, including complex and heterogeneous physical phenomena that can only be described through nonlinear differential equations. Thanks to its co-simulation interface, 3D-ICE 3.0 can be extended to support new heat sink technologies designed at a high abstraction level through equation-based modeling languages. The end goal of 3D-ICE 3.0 is to create an ecosystem of interoperable chip and sink models addressing the simulation needs of a heterogeneous landscape. The included air and water heat sink models showcase the capabilities of the simulator, supporting for the first time variable coolant flow rate, a feature that we expect will be soon exploited for the design of thermal control policies [50].

References

[1] *Intel Xeon Platinum 9282*. Accessed: Apr. 23, 2021. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/194146/intel-xeon-platinum-9282-processor-77m-cache-2–60-ghz.html

[2] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2011, pp. 365–376.

[3] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, pp. 1–6.

[4] J. Gullbrand, M. J. Luckeroth, M. E. Sprenger, and C. Winkel, "Liquid cooling of compute system," *J. Electron. Packag.*, vol. 141, no. 1, Mar. 2019, Art. no. 010802.

[5] K. Zhang, Y. Zhang, J. Liu, and X. Niu, "Recent advancements on thermal management and evaluation for data centers," *Appl. Thermal Eng.*, vol. 142, pp. 215–231, Sep. 2018.

[6] A. Seuret, A. Iranfar, M. Zapater, J. Thome, and D. Atienza, "Design of a two-phase gravity-driven micro-scale thermosyphon cooling system for high-performance computing data centers," in *Proc. 17th IEEE ITherm*, 2018, pp. 587–595.

[7] A. A. Andreev *et al.*, "PowerCool: Simulation of cooling and powering of 3D MPSoCs with integrated flow cell arrays," *IEEE Trans. Comput.*, vol. 67, no. 1, pp. 73–85, Jan. 2018.

[8] G. Bulman *et al.*, "Superlattice-based thin-film thermoelectric modules with high cooling fluxes," *Nat. Commun.*, vol. 7, no. 1, Jan. 2016, Art. no. 10302.

[9] F. Kaplan, M. Said, S. Reda, and A. Coskun, "LoCool: Fighting hot spots locally for improving system energy efficiency," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 4, pp. 895–908, Apr. 2020.

[10] A. Iranfar *et al.*, "Thermal characterization of next-generation workloads on heterogeneous MPSoCs," in *Proc. Int. Conf. Embedded Comput. Syst. Archit. Model. Simulat. (SAMOS)*, 2017, pp. 286–291.

[11] S. Pagani *et al.*, "TSP: Thermal safe power—Efficient power budgeting for many-core systems in dark silicon," in *Proc. Int. Conf. Hardw. Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2014, pp. 1–10.

[12] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2010, pp. 463–470.

[13] 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 Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[14] S. Ladenheim, Y. Chen, M. Mihajlović, and V. F. Pavlidis, "The MTA: An advanced and versatile thermal simulator for integrated systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3123–3136, Dec. 2018.

[15] T. L. Bergman, F. P. Incropera, D. P. DeWitt, and A. S. Lavine, *Fundamentals of Heat and Mass Transfer*. London, U.K.: Wiley, 2011.

[16] *The Functional Mock-up Interface (FMI) Standard*. Accessed: Apr. 23, 2021. [Online]. Available: https://fmi-standard.org/

[17] *Tools Supporting the Functional Mock-up Interface (FMI) Standard*. Accessed: Apr. 23, 2021. [Online]. Available: https://fmi-standard.org/tools/

[18] Y. Yang, C. Zhu, Z. Gu, L. Shang, and R. P. Dick, "Adaptive multi-domain thermal modeling and analysis for integrated circuit synthesis and design," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, 2006, pp. 575–582.

[19] Y.-M. Lee, C.-W. Pan, P.-Y. Huang, and C.-P. Yang, "LUTSim: A look-up table-based thermal simulator for 3-D ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 8, pp. 1250–1263, Aug. 2015.

[20] W. Yu, T. Zhang, X. Yuan, and H. Qian, "Fast 3-D thermal simulation for integrated circuits with domain decomposition method," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 12, pp. 2014–2018, Dec. 2013.

[21] S. Ladenheim, Y.-C. Chen, M. Mihajlović, and V. Pavlidis, "IC thermal analyzer for versatile 3-D structures using multigrid preconditioned Krylov methods," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2016, pp. 1–8.

[22] Q. Xie, M. J. Dousti, and M. Pedram, "Therminator: A thermal simulator for smartphones producing accurate chip and skin temperature maps," in *Proc. Int. Symp. Low Power Electron. Design*, 2014, pp. 117–122.

[23] Z. Hassan, N. Allec, F. Yang, L. Shang, R. P. Dick, and X. Zeng, "Full-spectrum spatial–temporal dynamic thermal analysis for nanometer-scale integrated circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 12, pp. 2276–2289, Dec. 2011.

[24] *Hotspot Differential Equation Integration Method*. Accessed: Apr. 23, 2021. [Online]. Available: https://github.com/uvahotspot/HotSpot/blob/master/RCutil.c#L286

[25] F. Terraneo, A. Leva, and W. Fornaciari, *Harnessing Performance Variability in Embedded and High-performance Many/Multi-core Platforms: A Cross-layer Approach* (Event-Based Thermal Control for High Power Density Microprocessors). Cham, Switzerland: Springer, 2019, pp. 107–127.

[26] Peng Li, L. T. Pileggi, M. Asheghi, and R. Chandra, "IC thermal simulation and modeling via efficient multigrid-based approaches," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 9, pp. 1763–1776, Sep. 2006.

[27] F. E. Cellier and E. Kofman, *Continuous System Simulation*. Boston, MA, USA: Springer, 2006.

[28] R. Winterton, "Where did the Dittus and Boelter equation come from?" *Int. J. Heat Mass Transf.*, vol. 41, nos. 4–5, pp. 809–810, 1998.

[29] R. Temam, *Navier-Stokes Equations: Theory and Numerical Analysis*, vol. 343. Providence, RI, USA: Amer. Math. Soc., 2001,

[30] J. Gmehling, M. Kleiber, B. Kolbe, and J. Rarey, *Chemical Thermodynamics for Process Simulation*. Weinheim, Germany: Wiley, 2019.

[31] A. Benveniste, B. Caillaud, and M. Malandain, "The mathematical foundations of physical systems modeling languages," *Annu. Rev. Control*, vol. 50, pp. 72–118, Apr. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578820300547

[32] M. Nesarajah and G. Frey, "Modeling of a heat pipe for using in thermoelectric energy harvesting systems," in *Proc. 3rd Int. Congr. Energy Efficiency Energy Related Mater. (ENEFM)*, 2017, pp. 183–190.

[33] C. Junior, C. Richter, W. Tegethoff, N. Lemke, and J. Köhler, "Modeling and simulation of a thermoelectric heat exchanger using the object-oriented library til," in *Proc. 6th Int. Modelica Conf.*, 2008, pp. 437–445.

[34] P. Li, Y. Li, and J. E. Seem, "Modelica-based dynamic modeling of a chilled-water cooling coil," *HVAC R Res.*, vol. 16, no. 1, pp. 35–58, 2010.

[35] F. Casella and A. Leva, "Modelling of thermo-hydraulic power generation processes using modelica," *Math. Comput. Model. Dyn. Syst.*, vol. 12, no. 1, pp. 19–33, 2006.

[36] F. Casella and C. Richter, "Externalmedia: A library for easy re-use of external fluid property code in modelica," in *Proc. 6th Int. Modelica Conf.*, 2008, pp. 157–161.

[37] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey," *ACM Comput. Surveys*, vol. 51, no. 3, p. 49, 2018.

[38] M. Busch, "Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error," *J. Appl. Math. Mech. Zeitschrift Angewandte Mathematik Mechanik*, vol. 96, no. 9, pp. 1061–1081, 2016.

[39] W.-T. Chang, A. Kalavade, and E. A. Lee, *Effective Heterogenous Design and Co-Simulation*. Dordrecht, The Netherlands: Springer, 1996, pp. 187–212.

[40] F. Casella, F. Donida, and M. Lovera, "Beyond simulation: Computer aided control system design using equation-based object oriented modelling for the next decade," in *Proc. 2nd Int. Workshop Equ. Based Object-Oriented Lang. Tools*, 2008, pp. 35–45.

[41] P. Fritzson, "Modelica: Equation-based, object-oriented modelling of physical systems," in *Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems*. Cham, Switzerland: Springer, 2020, pp. 45–96.

[42] C. Andersson, "Methods and tools for co-simulation of dynamic systems with the functional mock-up interface," Ph.D. dissertation, Fac. Eng., Lund Univ., Lund, Sweden, 2016.

[43] E. Drenth *et al.*, "Consistent simulation environment with FMI based tool chain," in *Proc. 10 th Int. Modelica Conf.*, 2014, pp. 1277–1283.

[44] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation With Modelica 2.1*. London, U.K.: Wiley, 2004.

[45] W. Huang, K. Skadron, S. Gurumurthi, R. J. Ribando, and M. R. Stan, "Differentiating the roles of IR measurement and simulation for power and temperature-aware design," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Apr. 2009, pp. 1–10.

[46] F. Terraneo, A. Leva, and W. Fornaciari, "An open-hardware platform for MPSoC thermal modeling," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Cham, Switzerland: Springer, 2019, pp. 184–196.

[47] F. Terraneo, A. Leva, and W. Fornaciari. *Dataset: An Open-Hardware Platform for MPSoC Thermal Modeling*. Accessed: Apr. 23, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.3345878

[48] A. Bartolini, M. Cacciari, A. Tilli, and L. L. Benini, "Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 170–183, Jan. 2013.

[49] A. Leva, F. Terraneo, I. Giacomello, and W. Fornaciari, "Event-based power/performance-aware thermal management for high-density microprocessors," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 2, pp. 535–550, Mar. 2018.

[50] G. Agosta *et al.*, "The RECIPE approach to challenges in deeply heterogeneous high performance systems," *Microprocess. Microsyst.*, vol. 77, Sep. 2020, Art. no. 103185.

Federico Terraneo received the Master of Science degree in engineering of computing systems from the Politecnico di Milano, Milan, Italy, with the thesis "Control Based Design of OS Components," in 2011, and the Ph.D. degree in information technology with the thesis "Thermal and Energy Management Techniques for Multicore and Many-Core Systems," in 2015.

Since 2008, he has been the Original Designer, Main Developer, and Maintainer of Miosix (http://miosix.org), an operating system kernel targeting microcontroller-based embedded systems. His current research line focuses on thermal management for microprocessor-based computing systems and improving time determinism of distributed embedded systems. He has multidisciplinary research interests, including embedded systems and operating systems, with a lively interest in designing hardware and software components applying a controltheoretical methodology.

Alberto Leva (Member, IEEE) received the Laurea degree in electronic engineering (*summa cum laude*) from the Politecnico di Milano, Milan, Italy, in 1989.

In 1991, he joined the Dipartimeto di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, where he is currently a Professor of Automatic Control. His main research interests concern methods and tools for the automatic tuning of industrial controllers and control structures, process modeling, simulation and control, particularly within the object-oriented paradigm, object-oriented modeling and control of energy systems with particular reference to large grids, and advanced tools and methods for control education. In recent years, he has been concentrating on control and control-based design of computing systems, addressing in a system- and control-theoretical manner problems, such as scheduling, resource allocation, time synchronization in wireless sensor networks, thermal and power/performance management, performance-driven software adaptation, and service composition.

William Fornaciari (Senior Member, IEEE) received the Master degree in electronic engineering in 1989, and Ph.D. degree in computer enginnering in 1992.

He is now an Associate Professor with the Politecnico di Milano, Milan, Italy. He has published six books and around 300 papers in international journals and conferences, collecting six best paper awards and one certification of appreciation from IEEE. He holds three international patents on low power design and thermal management. He has been the coordinator of both FP7 and H2020 EU-Projects. His research interests include embedded and cyber–physical systems, energy-aware design of SW and HW, run-time management of resources, high-performance computing, design optimization, and thermal management of multimany cores.

Dr. Fornaciari won the HiPEAC Technology Transfer Award in 2016 and the Switch-To-Product Competition in 2019.

Marina Zapater (Member, IEEE) received the Ph.D. degree in electronic engineering from the Universidad Politécnica de Madrid, Madrid, Spain, in 2015.

She has been an Associate Professor with the School of Engineering and Management of Vaud (HEIG-VD), University of Applied Sciences Western Switzerland (HES-SO), Yverdon-les-Bains, Switzerland, since 2020, and a Research Associate with the Embedded System Laboratory, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland, since 2016. Her research interests include thermal, power and performance design and optimization of complex heterogeneous architectures, from embedded AI-enabled edge devices to high-performance computing processors; and energy efficiency in servers and data centers. In these fields, she has coauthored more than 50 papers in top-notch conferences and journals.

Dr. Zapater is a member of CEDA.

David Atienza (Fellow, IEEE) received the Ph.D. degrees in computer science and engineering from the Complutense University of Madrid, Madrid, Spain, and IMEC, Leuven, Belgium, in 2005.

He is a Professor of Electrical and Computer Engineering and the Head of the Embedded Systems Laboratory (ESL), Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland. He has coauthored more than 350 papers in peer-reviewed international journals and conferences, one book, and 12 patents. His research interests include system-level design methodologies for high-performance multiprocessor system-on-chip (MPSoC) and low power Internet-of-Things systems, including new 2-D/3-D thermal-aware design for MPSoCs and many-core servers, and edge AI architectures for wireless body sensor nodes and smart consumer devices.

Prof. Atienza received the 2020 ICCAD 10-Year Retrospective Most Influential Award, the DAC Under-40 Innovators Award in 2018, the IEEE TCCPS Mid-Career Award in 2018, the ERC Consolidator Grant in 2016, the IEEE CEDA Early Career Award in 2013, the ACM SIGDA Outstanding New Faculty Award in 2012, and the Faculty Award from Sun Labs at Oracle in 2011. He served as a DATE 2015 Program Chair and the DATE 2017 General Chair. He is an ACM Distinguished Member, and served as the IEEE CEDA President from 2018 to 2019.