Fast thermal simulation using SystemC-AMS

Fast Thermal Simulation using SystemC-AMS*

Yukai Chen, Sara Vinco, Enrico Macii and Massimo Poncino
Department of Control and Computer Engineering, Politecnico di Torino, Torino, Italy
name.surname@polito.it

ABSTRACT

Out of the many options available for thermal simulation of digital electronic systems, those based on solving an RC equivalent circuit of the thermal network are the most popular choice in the EDA community, as they provide a reasonable tradeoff between accuracy and complexity. HotSpot, in particular, has become the de-facto standard in these communities, although other simulators are also popular. These tools have many benefits, but they are relatively inefficient when performing thermal analysis for long simulation times, due to the occurrence of a large number of redundant computations intrinsic in the underlying models.

This work shows how a standard description language, namely SystemC and its analog and mixed-signal (AMS) extension, can be used to successfully simulate the equivalent thermal network,by achieving accuracy comparable to existing simulators, yet with much better performance. Results show that SystemC-AMS thermal simulation can outpace HotSpot simulation by 10X to 90X, with speedup improving as the size of the thermal network increases, and negligible estimation error. As a further advantage, the adoption of the same language to describe functionality and temperature allows the simultaneous simulation of both dimensions with no co-simulation overhead, thus enhancing the overall design flow.

Keywords

Thermal analysis; Thermal estimation; Simulation; SystemC-AMS

1. INTRODUCTION

Temperature is a critical dimension in embedded system design, as it heavily impacts performance, power consumption and reliability of the final system [2]. This has led to the use of a variety of tools for estimating temperature (both transient and steady-state), with the goal of enhancing the design flow with increased reliability and knowledge of the underlying physical mechanisms.

*This work was supported by the EC co-funded CONTREX (Design of embedded mixed-criticality CONTRol systems under consideration of EXtra-functional properties) project Grant Agreement FP7-ICT- 611146

GLSVLSI '16, May 18-20, 2016, Boston, MA, USA

DOI: http://dx.doi.org/10.1145/2902961.2902975

All these tools solve the same problem (i.e., a 2D or 3D heat diffusion equation) and basically differ in terms of their granularity and of the type of solver employed, with the latter aspect affecting their accuracy [7, 9, 10, 15, 16, 19]. In the EDA and computer architecture community, the de-facto standard for thermal simulation is HotSpot, a tool based on the circuit-equivalent of a thermal network, which achieves a good trade-off between granularity and accuracy [16]. HotSpot uses the chip floorplan and thermal package information to build an equivalent circuit description, that is solved over time on given traces of power dissipation.

A few approaches have been proposed to speed up the underlying equation solving mechanisms of HotSpot [8, 10, 17, 20]. However, all such solutions are relatively inefficient when performing thermal analysis for long simulation times, due to a large number of redundant computations intrinsic in the underlying models.

As a further limitation, all tools are designed to work standalone, thus making the integration with other embedded system design tools, *e.g.*, for functional or power simulation, extremely challenging. Executing the thermal simulators on traces produced offline by power simulation tools prevents the possibility of evaluating the mutual influence of temperature and power [5, 18]. This limitation is overcome by the construction of co-simulation frameworks, that integrate dedicated simulators through complex synchronization and data exchange mechanisms [3, 6]. However, co-simulation introduces a significant overhead, together with possible errors or timing misalignments.

This work shows how a standard functional description language, namely SystemC and its Analog and Mixed Signal extension (AMS) [1], can be successfully used to simulate the equivalent thermal network. The AMS extension provides electrical linear network constructs, thus allowing the straightforward implementation of the electrical circuit equivalent thermal model. The AMS solver proves to guarantee a high level of accuracy *w.r.t.* existing simulators, coupled with speedups of up to 90X *w.r.t.* HotSpot simulation, that increase with the size of the thermal network.

As a further advantage, the adoption of a functional language allows to simulate the thermal network simultaneously *w.r.t.* the system functional and power models, thus enhancing the overall embedded system desi gn flow.

2. BACKGROUND AND RELATED WORK

2.1 Thermal modeling

When referring to silicon chips, temperature estimation amounts to solving the heat diffusion equation in 3D, which is typically used to describe heat conduction in a chip and to calculate the temperature profile [11]:

$$\nabla^2 T + \frac{\dot{q}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \quad (1)$$

where T is the temperature, q is the heat flux (in W/m^2), k is thermal conductivity of the material (in $W/(mK)$), and $\alpha = \frac{k}{\rho c}$ is the thermal diffusivity, corresponding to the ratio between the thermal conductivity and volumetric heat capacity (density ρ times specific heat capacity c). $\nabla^2 T$ is the Laplacian of T and corresponds in 3D to $\frac{\partial T^2}{\partial x} + \frac{\partial T^2}{\partial y} + \frac{\partial T^2}{\partial z}$.

Equation 1 can be solved using numerical methods such as Finite Difference Method (FDM), Finite Element Method (FEM), or methods based on the Green function [13]. Both FDM and FEM methods discretize the entire chip according to some granularity and construct a system of linear equations, thus handling complicated material structures with different thermal properties in different layers. Methods based on Green function, conversely, provide a semi-analytical approach that analyzes only layers of interest. This reduces the problem size compared to FDM or FEM, but results in less accurate estimates due to the simplified two dimensional modeling of the thermal problem.

2.2 Related Work on Thermal Simulation

A number of methods for solving Equation 1 rely on the well-known *duality between thermal and electrical networks*, *i.e.*, they represent heat flow as a current passing through a thermal resistance and leading to a temperature difference, analogous to voltage. Such methods rely on existing circuit-level simulators like SPICE to solve the steady-state or transient voltage of the nodes of the equivalent circuit [9, 19].

Among the thermal simulators based on an electrical circuit equivalent, HotSpot [16] is the most popular one, particularly in the computer architecture and EDA communities. The equivalent circuit of the chip is built from a given floorplan and from the essential features of the thermal package. HotSpot solves Equation 1 at each time step by using an adaptive solver of Runge-Kutta equations, based on a given trace of power dissipation values. It can model both steady-state and transient cases, and it supports two levels of granularity (*i.e.*, block-level and grid-level) with obvious tradeoff between accuracy and speed. Other approaches try to reduce the overhead of the HotSpot equation solver by implementing some context-specific optimizations; for instance, the authors of [8] exploit periodicity in the power trace to speed up the solution of transient analysis, whereas [20] uses spatially and temporally adaptive techniques to reduce computation time.

SESCTherm [10] is a thermal modeling infrastructure based on finite-difference analysis, that adopts the same underlying equation solver as HotSpot, but adopts a mix of grid and block modes.

DTTEM [17] uses a similar interface as HotSpot (*i.e.*, conductance and capacitance matrices). The main difference between DTTEM and HotSpot lies in the temperature evaluation mechanism; by sampling power values at small and constant time intervals, transient temperature evaluation can be discretized, thus simplifying the solution of the heat transfer differential equation.

Except for the solutions based on SPICE simulations, all these approaches basically try to implement different strategies to speed up the solution of the differential heat diffusion equation by optimizing either time sampling or analysis granularity. However, they are relatively inefficient when performing thermal analysis for long simulation times, due to the occurrence of a large number of redundant computations intrinsic in the underlying models. Moreover, all approaches are based on pre-built power simulation traces, which implies that multiple runs correspond to multiple constructions of the equations and/or of the thermal network.

In this work we show that a standard circuit solver like the one provided by SystemC-AMS can result in comparable if not better speed/accuracy tradeoffs than those optimized solutions, while achieving the benefit of a concurrent simulation of functionality, power and temperature within a single simulation run.

2.3 Introduction to SystemC-AMS

SystemC-AMS extends SystemC with constructs for modelling analog and mixed-signal systems (AMS) [1]. To cover a wide variety of domains, SystemC-AMS defines three different abstraction levels, supporting different communication styles and representations *w.r.t.* the physical domain. *Timed Data-Flow* (TDF) models are scheduled statically by considering their producer-consumer dependencies in the discrete time domain. *Linear Signal Flow* (LSF) supports the modelling of continuous time through a library of pre-defined primitive modules (*e.g.*, integration, delay), each associated with a linear equation. Finally, the *Electrical Linear Network* (ELN) level models electrical networks through the instantiation of predefined primitives, *e.g.*, resistors or capacitors, associated with electrical equations.

SystemC-AMS is supported by an internal solver, that analyzes the ELN and LSF components to derive the equations modelling system behaviour. The equations are solved numerically over time to determine the evolution of system state, by adopting numerical solvers, *e.g.*, Euler and trapezoidal methods. The internal solver also guarantees that ELN descriptions are conservative, *i.e.*, the set of equations derived from ELN is extended with the application of conservation laws.

3. A SYSTEMC-AMS THERMAL SIMULATOR FOR SOC

The construction of the SystemC-AMS thermal model relies on electrical circuit equivalent models, as they allow to reduce a phenomenon that can not be natively represented in SystemC-AMS (*i.e.*, temperature) to linear networks, that have a dedicated abstraction level. The RC network is built by following state-of-the-art methods, and in particular the method used by HotSpot [16]. The novelty of the current work lies in the code generation process. HotSpot provides a stand-alone tool that explicitly solves circuit equations modeled as matrices. On the contrary, the proposed approach exploits the native support of SystemC-AMS for electric network primitives to map the RC network elements one-to-one to SystemC-AMS constructs (*e.g.*, resistors and capacitors). Circuit equations are then automatically derived and solved by the SystemC-AMS internal solver.

The resulting methodology is as depicted in Figure 1:

- *construction of the RC network* by reproducing the method used by HotSpot (Section 3.1);
- *SystemC-AMS code generation*, achieved by mapping the RC network elements to SystemC-AMS primitives (Section 2.3);
- simulation of the RC network by using the SystemC-AMS simulation kernel as a circuit solver (Section 3.3). Inputs are provided by a dedicated testbench, either (a) as *power dissipation traces* (á la HotSpot), or (b) by simulating the thermal model *in parallel w.r.t. functional and power models*.

3.1 Construction of the RC network

The algorithm to construct the RC network reproduces the method followed by HotSpot. In the current version of the methodology, we support only block-level simulation, *i.e.*, the simulator estimates one temperature value for each component. Finer granularity, *i.e.*, the grid-level of HotSpot, will be supported as part of future work. Both steady-state and transient simulation are supported.

Figure 2 exemplifies the application to a simple case study consisting of a core, a memory, a RF transceiver and a UART device.

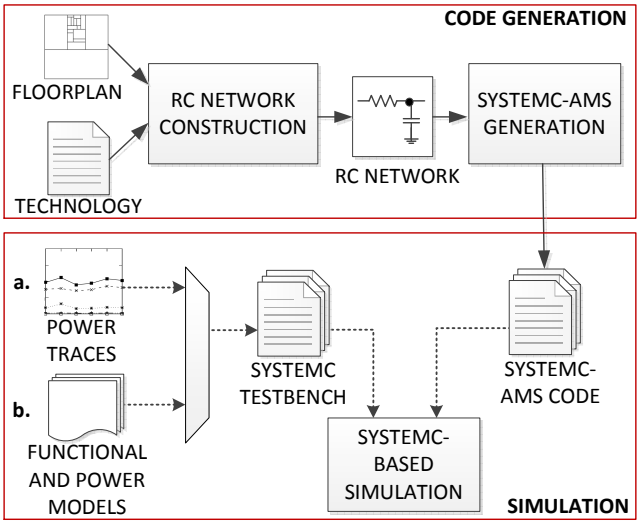


Figure 1: Proposed methodology for the construction of the SystemC-AMS thermal simulator.

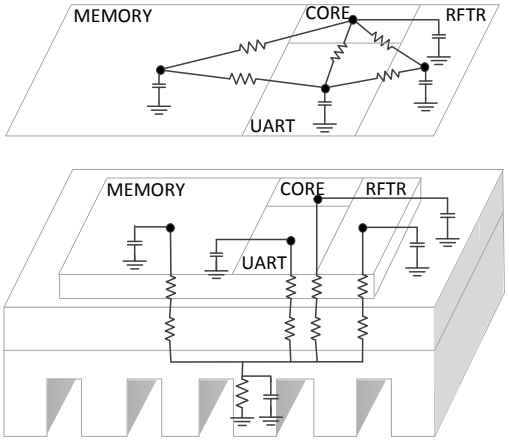


Figure 2: Example of electrical circuit equivalent model: the RC network is decomposed in a lateral model (top), representing heat transfers that occur horizontally between adjacent components, and a vertical model (bottom), that sketches the heat spread across the package layers.

The construction of the RC network heavily depends on the input information: chip floorplan, necessary to determine which components are adjacent, and technology information (*e.g.*, number of layers, materials, thermal characteristics).

Each chip component is mapped to a *RC network node*, whose current represents power consumption and whose voltage represents temperature. Additional nodes are used to represent the underlying package layers (*i.e.*, heat spreader and heat sink). Heat transfer flow between adjacent nodes is represented by a *resistor*, whose thermal resistance is proportional to the thickness of the material and inversely proportional to the cross-sectional area and to the thermal conductivity. Resistance values are stored in a matrix whose lines and columns are circuit nodes. If the RC network is used for transient simulation, *capacitors* are connected to each node, to capture the delay before a change in power determines a change in temperature. Thermal capacitance is proportional to both thickness and area, and it depends on the thermal capacitance per unit volume. Capacitance values are stored in a dedicated array.

3.2 SystemC-AMS code generation

The second step is the implementation of the computed RC network in SystemC-AMS. Figure 3 exemplifies the proposed approach by showing how the lateral model depicted on top of Figure 2 is implemented in SystemC-AMS[1].

Interface modeling.

The electrical circuit equivalent model is instantiated as a single SystemC module (`SC_MODULE`, line 1), encapsulating the entire RC network. The interface of the module is made of two ports for each chip component: one input port to gather the evolution of power consumption over time, and one output port to convey the corresponding value of temperature.

The flexibility of SystemC-AMS and its support for multiple levels of abstraction allows to decouple the semantics of the interface from the semantics of the actual behavior implementation. For this reason, the abstraction level adopted for ports is TDF, that determines a fixed timestep at which input ports are read, RC network is evaluated and output ports are updated. This reflects the behavior of HotSpot, that assumes that the power traces contain samples collected at fixed time steps. Ports are thus declared as `sca_tdf::sca_in` and `sca_tdf::sca_out` ports of type double (lines 2–3).

Body implementation.

The body of the `SC_MODULE` includes the implementation of the RC network, by reproducing the model computed as in Section 3.1. This is realized by adopting the ELN level of abstraction, that natively supports electrical network elements. The construction of the SystemC-AMS thermal model is thus a one-to-one mapping of circuit elements into SystemC-AMS primitives.

Each circuit node is implemented as a SystemC-AMS ELN node (`sca_node`, lines 6–7), despite of ground, that is represented with an ad-hoc primitive (`sca_node_ref`, line 5).

Resistors are mapped to instances of the `sca_r` primitive, that represents SystemC-AMS resistors (line 12). The resistance value is extracted from the resistance matrix computed in the previous methodology step (Section 3.1); *e.g.*, the snapshot of code in Figure 3 shows that the resistance modeling the heat flow between the core and the memory is 215.48Ω (lines 27–30).

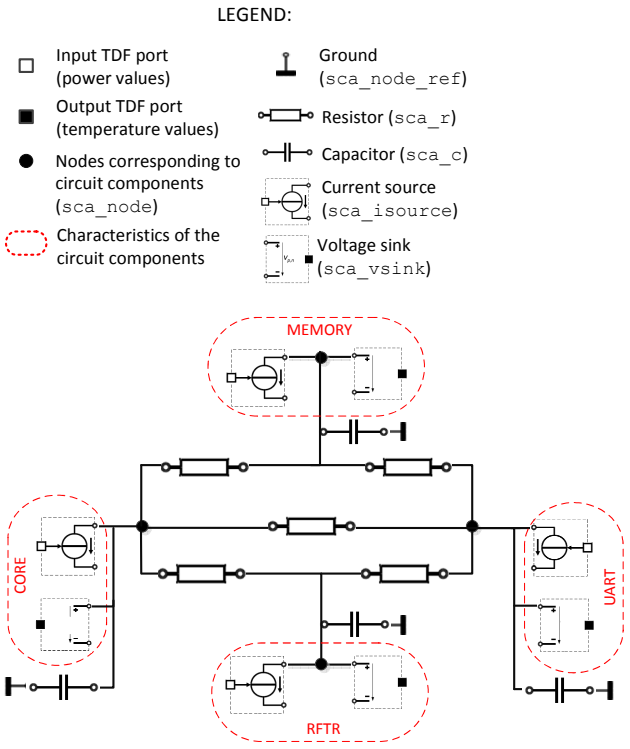
Capacitors are mapped to an instance of the `sca_c` primitive, that represents SystemC-AMS capacitors (line 11). The capacitance value is extracted from the computed capacitance array; *e.g.*, the core capacitance is 874 10^{-5}F (lines 23–26).

The input power ports are connected to the ELN circuit via current source primitives, that transform a numerical value into a current (`sca_isource` primitive, lines 9 and 15–18). In the pictorial representation in Figure 3, TDF ports are represented by the white square terminals of the primitive blocks.

Conversely, the temperature of each chip component is extracted through a voltage sink, that extrapolates a voltage value and makes it available on the output ports. Voltage sinks are represented with instances of the `sca_vsink` primitive, whose output terminal is connected to the corresponding output temperature port (the black square terminals, lines 10 and 19–22).

3.3 Stimuli Generation

[1]Note that this constitutes only a subset of the complete RC network. Symbols adopted for the primitives are as standardized in the SystemC-AMS standard [1].




```
1.  SC_MODULE (thermal_network){
2.    sca_tdf::sca_in<double> p_CORE ;
3.    sca_tdf::sca_out<double> t_CORE ;
4.    ...
5.    sca_eln::sca_node_ref gnd;
6.    sca_eln::sca_node node_CORE;
7.    sca_eln::sca_node node_MEMORY;
8.    ...
9.    sca_eln::sca_tdf::sca_isource* iCORE;
10.   sca_eln::sca_tdf::sca_vsink* vCORE;
11.   sca_eln::sca_c* c_CORE;
12.   sca_eln::sca_r* r_MEM_CORE;
13.   ...

14.  SC_CTOR(thermal_network){
15.   i_CORE = new sca_eln::sca_tdf::sca_isource("i_CORE");
16.   i_CORE->p(gnd);
17.   i_CORE->n(node_CORE);
18.   i_CORE->inp(p_CORE);

19.   v_CORE = new sca_eln::sca_tdf::sca_vsink("v_CORE");
20.   v_CORE->p(node_CORE);
21.   v_CORE->n(gnd);
22.   v_CORE->outp(t_CORE);

23.   c_CORE = new sca_eln::sca_c("c_CORE");
24.   c_CORE->p(node_CORE);
25.   c_CORE->n(gnd);
26.   c_CORE->value=0.00008741;

27.   r_MEM_CORE= new sca_eln::sca_r("r_MEM_CORE");
28.   r_MEM_CORE->p(node_MEMORY);
29.   r_MEM_CORE->n(node_CORE);
30.   r_MEM_CORE->value = 215.48122572;
31.   ...
32. } };
```

Figure 3: Example of implementation of the lateral model of Figure 2 in SystemC-AMS: implementation of the RC network with ELN primitives (left) and excerpt of SystemC-AMS code (right).

The final step is the generation of input stimuli. The SystemC-AMS thermal simulator is connected to a *testbench module* through a complementary interface: power ports are in output, and temperature ports are in input. The testbench generates stimuli over time for the thermal simulator, by adopting two possible strategies.

The first option is to load *dissipation power traces* from files previously generated by a power simulator (option *a* in Figure 1). The estimated temperature values are then dumped to a thermal trace file for future elaborations. This corresponds to the simulation semantics of all thermal simulators, including HotSpot.

The second option consists of simulating the RC network on *dynamically generated power information* (option *b* in Figure 1). In this scenario, the testbench wraps SystemC processes implementing power models, or functional models enriched with power information [4, 12]. This approach is made possible by the adoption of a functional language for thermal simulation. A major advantage is that power information is generated dynamically, depending on the evolution of functionality, and that the mutual effects of power and temperature can be reproduced at runtime.

An example of this strategy is sketched in Figure 4. The testbench includes both a functional model and a power model: the `pause` functional signal is used to determine the evolution of the power state machine, that dynamically updates the power values written in output to the thermal model.

It is important to note that this second scenario provides a novelty *w.r.t.* the current state-of-the-art. Simultaneous simulation is indeed achieved by simulating all aspects in a single simulation run and by adopting a single simulation kernel, *i.e.*, SystemC with its AMS extension. Comparing this solution with approaches such as [3, 6], we avoid the overhead and the criticality of building a co-simulation framework, thus achieving faster simulation and an enhanced guarantee of correctness.

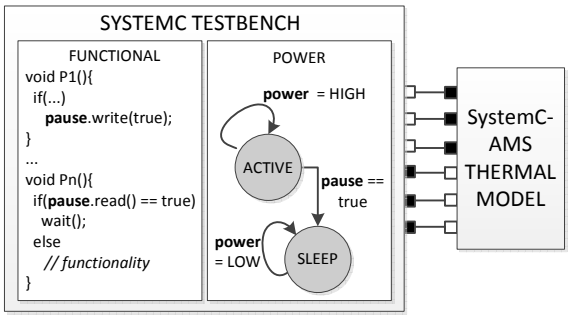


Figure 4: Example of how functional and power models can influence thermal simulation through simultaneous simulation.

4. EXPERIMENTAL RESULTS

4.1 Simulation Setup

The proposed methodology has been automated in a C++ tool, that takes in input two files, modeling floorplan and technology information. The tool builds the RC network and saves the generated code to a SystemC file, named by default `RCnetwork.h`. Furthermore, the tool generates the interface of the testbench module,

while its implementation is left to the designer. Table 1 summarizes the technology values. Air is always set to a fixed ambient temperature of 40^{o}C. Convection capacitance *w.r.t.* ambient is 140.4 $\frac{J}{K}$, and convection resistance is 0.1 $\frac{K}{W}$.

Layers	Thickness	Thermal Conductivity
Die	1.5e-04 m	100.0 W/m-K
TIM	2.0e-05 m	4.000 W/m-K
Heat-spreader	1.0e-03 m	400.0 W/m-K
Heat-sink	6.9e-03 m	400.0 W/m-K

Table 1: Technology parameters.

4.2 Case Studies

To compare our methodology to HotSpot, we used five benchmarks of different sizes and complexity: the example in Figure 2 (benchmark 1), two built-in examples of HotSpot (benchmark 2 and 3), and two synthetic benchmarks (4 and 5). The latter two do not correspond to any functionality, and are only used to prove effectiveness and scalability of the proposed simulation approach. The benchmarks have been generated by replicating a number of cores, caches and memories, with typical power consumption traces. The corresponding floorplans have been derived by using the HotSpot thermal-aware floorplanning tool.

To ensure a fair comparison, we fed both HotSpot and our tool with the same power traces and floorplan and technology files.

4.3 Correctness of the Computed RC Network

To test the correctness of our approach, we compared the RC network built by our approach for each benchmark with the one built by HotSpot. Table 2 reports the main characteristics of the RC networks.

Since our RC network construction algorithm is based on HotSpot, it is no surprise that the RC networks are identical, in terms of both number of nodes, resistors and capacitors. This guarantees that both SystemC-AMS and HotSpot solve the same equations, and that any inaccuracy lies in the solver, rather than in the constructed thermal model.

Benchmark	Components (#)	Nodes (#)	Resistors (#)	Capacitors (#)
1	4	28	64	28
2	18	84	288	84
3	30	132	442	132
4	40	172	586	172
5	86	356	1,206	356

Table 2: Characteristics of the computed RC networks.

4.4 Simulation Time

To evaluate the temporal performance of SystemC-AMS simulations, we compare separately the time to generate the RC thermal model and the time to simulate the network. Table 3 reports *generation times*, and shows that the time necessary to build the thermal model is similar for HotSpot and SystemC-AMS, since both the approaches rely on the same algorithm for RC network construction. The time is slightly higher for our approach, since it requires a further step to map the resistor and capacitor matrices to SystemC-AMS primitives, and to print the generated code onto a file. To overcome this limitation, we could avoid the dump of the RC on a file and rely on some interprocess communication facility to exchange the data structure representing the netlist with SystemC-AMS. In any case, generation times are truly negligible, as shown in Table 3.

Benchmark	SystemC-AMS (s)	HotSpot (s)
1	0.002	0.001
2	0.004	0.001
3	0.011	0.002
4	0.016	0.003
5	0.120	0.020

Table 3: Comparison of RC Network Generation Time.

Table 4 completes the analysis with the most relevant information, i.e., the comparison of *simulation times*. Data refers to a 1000ms trace with timestep 1ms. SystemC-AMS proves to be much faster than HotSpot, with a speedup of up to 96X. The interesting aspect is that speedup scales linearly with the size of the netlist. This proves that the improved performance is due to the SystemC-AMS simulation kernel, that handles more efficiently the RC network equations. Note that the achieved simulation speedup compensates for the slightly longer generation time (Table 3). These results allow us to claim that our approach is competitive also *w.r.t.* optimized variants of HotSpot, that achieve speedups around 4X over standard HotSpot [21], thus far lower than our experienced performance.

Benchmark	SystemC-AMS (s)	HotSpot (s)	Speedup
1	0.101	1.472	14.55
2	0.536	17.446	32.55
3	0.983	40.609	41.31
4	1.425	72.591	50.94
5	3.343	321.397	96.14

Table 4: Comparison of Simulation Time.

4.5 Accuracy

To prove the accuracy of our simulator, we determine the relative error *w.r.t.* HotSpot, by analyzing the traces of each component. The relative error is computed as in Equation 2:

$$Err_{relative} = (T_{AMS} - T_{HotSpot})/(T_{HotSpot} - T_{Ambient}) \quad (2)$$

We first compared SystemC-AMS and HotSpot on *steady-state scenarios*. The input power traces consist of a single power value per component, corresponding to the average power dissipation of the component over time. SystemC-AMS proved to be extremely accurate, as we got a 0% error on all benchmarks. This can be explained by the similar underlying solvers, and by the fact that steady-state evaluation does not require iterations, thus avoiding the accumulation of approximations.

Case Study	Avg. error (%)
1	0.034
2	0.052
3	0.140
4	0.160
5	0.015

Table 5: Accuracy of transient temperature estimation.

The *transient scenario* is more interesting, as the RC network includes also capacitors and multiple iterations are necessary to compute the thermal trace. Table 5 shows that the average error committed on all benchmarks is far below 1%. This proves that the SystemC-AMS solver is extremely accurate *w.r.t.* the HotSpot simulation kernel. Figure 5 proves the high level of accuracy by showing that the temperature curves computed by SystemC-AMS and HotSpot for benchmark 1 are overlapped (top), and that the

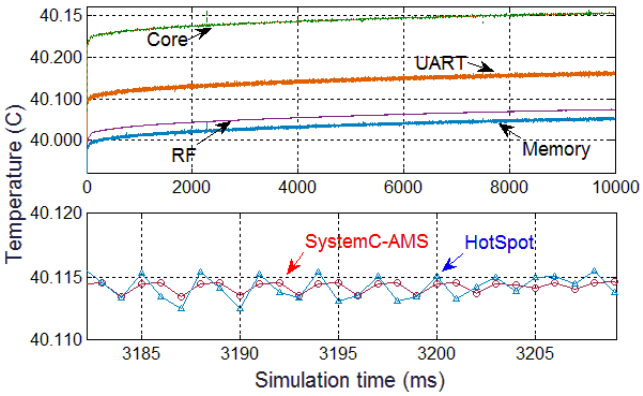


Figure 5: Transient temperature profile from SystemC-AMS and HotSpot applied to benchmark 1 (top) and zoomed view of the memory transient temperature profiles (bottom).

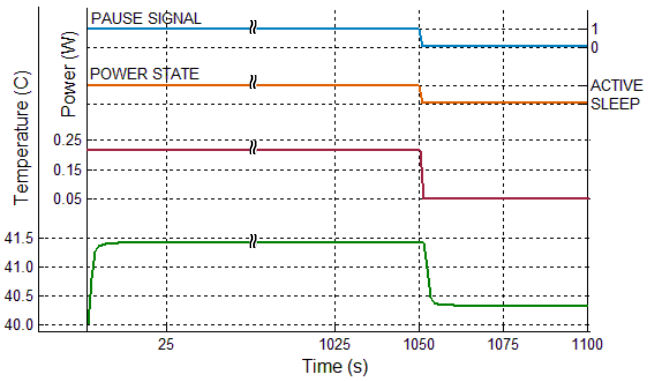


Figure 6: Effect of simultaneous simulation of functional and power models with the SystemC-AMS thermal model.

error becomes visible only by zooming in on very small time windows (bottom).

By zooming in the curves, we observed that the maximum error always happens at inflection points of the temperature profiles. This is caused by the electrical characteristics of SystemC-AMS capacitors and resistors, that result in slightly different time constants. The inspection of the traces suggests however that the less abrupt transitions resulting by SystemC-AMS simulation are more realistic than the discontinuous sawtooth profile computed by the solver in HotSpot.

4.6 Simultaneous simulation with functional and power models

As a final experiment, we implemented a scenario in which the SystemC-AMS thermal simulation of benchmark 1 (Figure 2) is run concurrently with functional and power models. To this extent, we enriched the testbench by implementing the functional and power models sketched in Figure 4 as SystemC processes. Our goal was to dynamically derive power information for the digital core from its functional evolution [14], to determine the influence of the executed application on the thermal profile.

Figure 6 traces the evolution of the resulting simulation. As soon as the `pause` signal is set to 0, the power model reacts and performs the transition from $ACTIVE$ to $SLEEP$, thus lowering power dissipation from 0.2314W to 0.0500W. This is detected by the thermal simulator, that gradually decreases the core temperature from 41.42°C to 40.33°C. Overall simulation lasted 6.302s and was performed in a single run with the sole support of the SystemC simulation kernel. Building the same scenario with HotSpot, or with any other custom thermal simulator, would have required the construction of a co-simulation infrastructure with an architectural or a circuit simulator.

5. CONCLUSIONS

This work demonstrated that SystemC-AMS can be used to successfully simulate thermal evolution by modeling the equivalent thermal network through ELN primitives without the need of implementing custom circuit simulators. Experimental results showed that our SystemC-AMS-based approach reaches a high level of accuracy *w.r.t.* HotSpot, with an average error lower that 1% on all tested benchmarks, while achieving speedup of up to 90X, that scales linearly with the size of the thermal network. Given this latter nice scalability feature, we are confident that the speedup will be even larger for grid-level simulation, where the number of electrical elements and circuit nodes is much larger. This will be our future work on this subject.

6. REFERENCES

[1] Accellera. *SystemC-AMS*. www.systemc-ams.org.

[2] A. H. Ajami, K. Banerjee, and M. Pedram. Modeling and analysis of nonuniform substrate temperature effects on global ULSI interconnects. *IEEE TCAD*, 24(6):849–861, 2005.

[3] B. Beckmann, Y. Eckert, et al. A comprehensive timing, power, thermal, and reliability model for exascale node architectures. In *Proc. of DOE MODSIM*, 2013.

[4] L. Benini, R. Hodgson, and P. Siegel. System-level power estimation and optimization. In *Proc. of ACM/IEEE ISLPED*, pages 173–178, 1998.

[5] N. Hatami, R. Baranowski, et al. Multilevel simulation of nonfunctional properties by piecewise evaluation. *ACM TODAES*, 19(4):37:1–37:21, 2014.

[6] M. Hsieh, K. Pedretti, J. Meng, et al. SST + Gem5 = a scalable simulation infrastructure for high performance computing. In *Proc. of ACM SIMUTOOLS*, pages 196–201, 2012.

[7] T. Kemper, Y. Zhang, Z. Bian, and A. Shakouri. Ultrafast temperature profile calculation in IC chips. In *IEEE THERMINIC*, pages 133–137, 2006.

[8] P. Liu, Z. Qi, H. Li, L. Jin, W. Wu, S. Tan, and J. Yang. Fast thermal simulation for architecture level dynamic thermal management. In *IEEE ICCAD*, pages 639–644, 2005.

[9] W. Liu, A. C. A., A. Macii, et al. Layout-driven post-placement techniques for temperature reduction and thermal gradient minimization. *IEEE TCAD*, 32(3):406–418, 2013.

[10] J. Nayfach and J. Renau. SOI, interconnect, package, and mainboard thermal characterization. *IEEE ISLPED*, pages 327–330, 2009.

[11] M. Ozisik. *Boundary Value Problems of Heat Conduction*. Oxford University Press, 1968.

[12] X. Pan, J. Molina, and C. Grimm. Modeling power consumption at system-level for design of power integrity-aware AMS-circuits. In *Proc. of ECSI/IEEE FDL*, pages 1–8, 2015.

[13] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in VLSI circuits: Principles and methods. *Proc. of the IEEE*, 94(8):1487–1501, 2006.

[14] S. Rhoads. Plasma CPU Core, 2001. opencores.org.

[15] K. Skadron, M. Stan, W. Huang, et al. Temperature-aware computer systems: Opportunities and challenges. *IEEE Micro*, 23(6):52–61, 2003.

[16] K. Skadron, M. R. Stan, B. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture: Extended discussion and results. Technical Report CS-2003-08, Univ. of Virginia Dept. of CS, 2003.

[17] L. Thiele, L. Schor, H. Yang, and I. Bacivarov. Thermal-aware system analysis and software synthesis for embedded multi-processors. In *ACM/EDAC/IEEE DAC*, pages 268–273, 2011.

[18] A. Viehl, B. Sander, O. Bringmann, and W. Rosenstiel. Integrated requirement evaluation of non-functional system-on-chip properties. In *ECSI/IEEE FDL*, pages 105–110, 2008.

[19] T.-Y. Wang and C. Chen. SPICE-compatible thermal simulation with lumped circuit modeling for thermal reliability analysis based on modeling order reduction. In *IEEE ISQED*, pages 357–362, 2004.

[20] Y. Yang, Z. Gu, C. Zhu, R. Dick, and L. Shang. ISAC: Integrated space-and-time-adaptive chip-package thermal analysis. *IEEE TCAD*, 26(1):86–99, 2007.

[21] A. Ziabari, E. Ardestani, J. Renau, and A. Shakouri. Fast thermal simulators for architecture level integrated circuit design. In *IEEE SEMI-THERM*, pages 70–75, 2011.