HW-SW Emulation Framework for Temperature-Aware Design in MPSoCs

DAVID ATIENZA and PABLO G. DEL VALLE
DACYA—UCM and LSI—EPFL
GIACOMO PACI, FRANCESCO POLETTI, and LUCA BENINI
DEIS—University of Bologna
GIOVANNI DE MICHELI
LSI—Ecole Polytechnique Federale de Lausanne (EPFL)
and
JOSE M. MENDIAS and ROMAN HERMIDA
DACYA—Complutense University of Madrid (UCM)

New tendencies envisage multiprocessor systems-on-chips (MPSoCs) as a promising solution for the consumer electronics market. MPSoCs are complex to design, as they must execute multiple applications (games, video) while meeting additional design constraints (energy consumption, time-to-market). Moreover, the rise of temperature in the die for MPSoCs can seriously affect their final performance and reliability. In this article, we present a new hardware-software emulation framework that allows designers a complete exploration of the thermal behavior of final MPSoC

An initial version of this article was presented in the 2006 *IEEE/ACM Design Automation Conference (DAC)*. This extended version presents several new contributions:

(i) Extensions of the description of the developed thermal library and emulated SoC.
(ii) Complete overview of related work on MPSoC modeling, testing and thermal-aware design.
(iii) Study of different thermal library models to characterize final MPSoC designs.
(iv) Illustration of the application of the tool with different floorplans.
(v) Evaluation of the impact of employing different packaging technologies in target MPSoCs.

This work was partially supported by the Spanish Government Research Grants TIN2005-5619 and FPU AP2005-0073, the Swiss NSF Research Grant 20021-109450/1, and the SRC Contract 1188.
Authors' addresses: D. Atienza (contact author), P. G. del Valle, J. M. Mendias, R. Hermida, Departamento de Arquitectura de Computadores y Automatica (DACYA), Universidad Complutense de Madrid, Ciudad Universitaria—28040 Madrid, Spain; email: datienza@dacya.ucm.es; G. Paci, F. Poletti, L. Benini, Dipartimento Elettronica Informatica e Sistemistica (DEIS), Universita di Bologna, Via Zamboni, 33-40126 Bologna-Partita IVA: 01131710376, Italy; G. de Micheli, Laboratoire des Systemes Integres (LSI), Ecole Polytechnique Federale de Lausanne, EPFL-IC-ISIM-LSI Station 14, Lausanne, Switzerland.

designs early in the design flow. The proposed framework uses FPGA emulation as the key element to model hardware components of the considered MPSoC platform at multimegahertz speeds. It automatically extracts detailed system statistics that are used as input to our software thermal library running in a host computer. This library calculates at runtime the temperature of on-chip components, based on the collected statistics from the emulated system and final floorplan of the MPSoC. This enables fast testing of various thermal management techniques. Our results show speedups of three orders of magnitude compared to cycle-accurate MPSoC simulators.

Categories and Subject Descriptors: C.3 [**Computer System Organization**]: Special-Purpose and Application-Based Systems—*Real-time and embedded systems*; B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

General Terms: Design, Measurement, Performance

Additional Key Words and Phrases: Thermal-aware design, FPGA, emulation, MPSoC, temperature

ACM Reference Format:

Atienza, D., del Valle, P. G., Paci, G., Poletti, F., Benini, L., de Micheli, G., Mendias, J. M., and Hermida, R. 2007. HW-SW emulation framework for temperature-aware design in MPSoCs. ACM Trans. Des. Autom. Electron. Syst. 12, 3, Article 26 (August 2007), 26 pages. DOI = 10.1145/1255456.1255463 http://doi.acm.org/10.1145/1255456.1255463

1. INTRODUCTION

An increasing number of multimedia services (e.g., multiview video or multiband wireless protocols) are being implemented on embedded consumer electronics, thanks to the fast evolution of process technology. These new embedded systems demand complex multiprocessor designs to meet their real-time processing requirements while respecting other critical embedded design constraints, such as low energy consumption or reduced implementation size. Moreover, the consumer market is reducing more and more their time-to-market and price [Jerraya and Wolf 2005], which does not permit complete redesigns of such multicore systems on a per-product basis. Thus, *multiprocessor systems-on-chips (MPSoCs)* have been proposed as a promising solution for this context, since they are single-chip architectures consisting of complex integrated components that communicate with each other at very high speeds [Jerraya and Wolf 2005]. Nevertheless, one of their main design challenges is the fast exploration of multiple *hardware (HW)* and *software (SW)* implementation alternatives with accurate estimations of performance, energy, and power to tune the MPSoC architecture in an early stage of the design process. In addition, it has been recently outlined that the problem of temperature rise in the components of future MPSoCs [Skadron et al. 2004] will increase further their system integration complexity.

With an objective of exploring the HW-SW interaction, several MPSoC simulators have been proposed, both at transaction and cycle-accurate levels, using *hardware description languages (HDLs)* and SystemC [Benini et al. 2005; Braun et al. 2003; CoWare 2004]. Also, recent SW tools can be added to them to evaluate in detail the thermal pressure in on-chip components based on runtime power consumption and floorplanning information of the final MPSoCs [Skadron et al. 2004]. Nevertheless, although these complex combined SW environments achieve accurate estimations of the system with thermal analysis, they are very limited in performance (approximately 10–100 KHz) due to signal management overhead. Thus, such environments cannot be used to analyze MPSoC solutions with complex embedded applications and realistic inputs of the final working environment to cover the variations in data loads at runtime. Moreover, higher abstraction-level simulators attain faster simulation speeds, but at the cost of a significant loss in accuracy. Hence, they are not suitable for fine-grained architectural tuning or thermal modeling.

One solution for the speed problems of cycle-accurate simulators is HW emulation. Various MPSoC emulation frameworks have been proposed [Cadence 2005; Heron Engineering 2004; Aptix 2003]. However, they are usually very expensive for embedded design (between $100K and $1M). Moreover, they are not flexible enough for MPSoC architecture exploration, since they mainly aim at large MPSoCs prototyping or SW debugging. Typically, the baseline architectures (e.g., processing cores or interconnections) are proprietary, not permitting internal changes. Furthermore, to the best of our knowledge, no flexible interconnection interfaces between HW emulation and current thermal SW libraries exist today. Thus, thermal effects can only be verified in the last phases of the design process, when the final components have been already developed, which can produce large overheads in system integration due to cores, as well as MPSoC architecture redesigns if any problem is discovered at that moment.

In this article we present a new HW-SW *field-programmable gate array (FPGA)*-based emulation framework that allows designers to explore a wide range of design alternatives for complete MPSoC systems at cycle-accurate levels, while characterizing their thermal behavior at a very fast speed (i.e., 100 MHz) with respect to MPSoC architectural simulators. First, MPSoC HW components are mapped on an FPGA to extract a large range of critical statistics from three key architectural levels of MPSoC systems (i.e., processing cores, the memory subsystem, and interconnection mechanisms), while real-life applications are executed. Second, this runtime information is sent through a flexible interface (using a standard Ethernet connection) to a configurable thermal model SW tool, running on a host PC, which evaluates at the same speed as the emulation executes the thermal behavior of the final MPSoC design, and returns this information to the FPGA emulating it. This final step enables testing runtime temperature management strategies in real time. Our results illustrate that the proposed HW-SW framework achieves detailed cycle-accurate reports with speedups of three orders of magnitude compared to state-of-the-art cycle-accurate MPSoC simulators. Moreover, our experiments indicate the benefit of the proposed framework to study the importance of packaging floorplan features in MPSoC designs.

The remainder of the article is organized as follows. In Section 2, we overview related work on MPSoC modeling, testing, and thermal-aware design. In Section 3, we present the configurable architecture of the emulated MPSoCs. In Section 4, we explain the used HW statistics extraction subsystem. In Section 5 we describe the configurable thermal SW libraries used. In Section 6, we detail how the HW-SW emulation process of MPSoC architectures is performed. In

Section 7, we illustrate the speed and versatility of our thermal emulation tool for MPSoC designers. Finally, we draw our conclusions in Section 8.

2. RELATED WORK

It is widely accepted that MPSoCs represent a promising solution for forthcoming complex embedded systems [Jerraya and Wolf 2005]. This has spurred research on modeling and prototyping MPSoC designs, using both HW and SW.

From the SW viewpoint, solutions have been suggested at different abstraction levels, enabling tradeoffs between simulation speed and accuracy. First, fast analytical models have been proposed to prune very distinct design options using high-level languages (e.g., C or C++) [Braun et al. 2003]. Also, full-system simulators, like Symics [Magnusson et al. 2002] and others, have been developed for embedded SW debugging and can reach megahertz speeds, but are not able to accurately capture performance and power effects (e.g., at the interconnection level) depending on the cycle-accurate behavior of the HW. Second, transaction-level modeling in SystemC, in academic [Paulin et al. 2002] and industrial context [CoWare 2004; ARM 2002] has enabled more accuracy in system-level simulation at the cost of sacrificing simulation speed (about 100–200 KHz). Such speeds render unfeasible the testing of large systems due to overly long simulation times, conversely to the proposed emulation framework. Moreover, in most cases SW simulations are only limited to a number of proprietary interfaces (e.g., AMBA [ARM 2004a] or Lisatek [CoWare 2004]). Finally, important research has been done to obtain cycle-accurate frameworks in SystemC or HDL languages. Companies have developed cycle-accurate simulators using postsynthesis libraries from HW vendors [Mentor Graphics 2003; Synopsys 2003]. However, their simulation speeds (10–50 KHz) are unsuitable for very complex MPSoC exploration. In the academic context, the MPARM SystemC framework [Benini et al. 2005] is a complete simulator for system exploration, since it includes cycle-accurate cores, complex memory hierarchies (e.g., caches, scratch pads), and interconnects, like AMBA or networks-on-chip (NoC). It can extract reliable energy and performance figures, but its major shortcoming is again its simulation speed (120 KHz in a P-4 at 2.8 GHz).

An important alternative to MPSoC prototyping and validation is HW emulation. In industry, one of the most complete sets of statistics is provided by Palladium II [Cadence 2005], which can accommodate very complex systems (i.e., up to 256 Mgate). However, its main disadvantages are its operation frequency (approximately 1.6 MHz) and cost (around $1 million). Then, ASIC integrator [ARM 2004a] is much faster for MPSoC architectural exploration. Nevertheless, its major drawback is its limitation to only up to five ARM-based cores and only AMBA interconnects. The same exploration limitation of proprietary cores occurs with Heron SoC emulation [Heron Engineering 2004]. Other relevant industrial emulation approaches are System Explore [Aptix 2003] and Zebu-XL [Emulation and Verification Engineering 2005], both based on multi-FPGA emulation in the order of MHz. They can be used to validate intellectual property blocks, but are not flexible enough for fast MPSoC design exploration or detailed statistics extraction. In the academic world, a relatively complete up-to-date emulation platform for exploring MPSoC alternatives is TC4SOC [Nava et al. 2005]. It uses a proprietary 32-bit VLIW core and enables exploration of interconnects by using an FPGA to reconfigure the *network interfaces (NIs)*. However, it does not enable detailed extraction of statistics and performing thermal modeling at the other two architectural levels we propose, namely, memory hierarchy and processing cores. Last, an interesting approach that uses FPGA prototyping to speed-up coverification of pure SW simulators is described in Nakamura et al. [2004]. In this case, the FPGA part is synchronized on a cycle-by-cycle basis with the C/C++ SW part by using an array of shared registers in the FPGA that can be accessed by both sides. This work shows a final speed for the combined framework of 1 MHz, outlining the potential benefits of combined HW-SW frameworks which we fully exploit in this approach to reach an MPSoC emulation speed of 100 MHz.

Regarding thermal modeling, Skadron et al. [2004] presented a thermal/power model for superscalar architectures. It can predict the temperature variations between different components of a processor and show the subsequent increased leakage power and reduced performance. Additionally, Su et al. [2003] investigated the impact of temperature and voltage variation across the die of an embedded core. Their results show that the temperature can vary by around 13.6 degrees across the die. Also, in López-Buedo et al. [2000] the temperature of FPGAs that are used as reconfigurable computers is measured using ring oscillators which can be dynamically inserted, moved, or eliminated. This empirical measurement method is interesting, yet only applicable to FPGAs that are as target devices. Our method alternatively aims at estimating the temperature of integrated circuits implementing MPSoC designs. Nevertheless, all of these works clearly prove the importance of hot spots in high-performance and reconfigurable systems, as well as the need for temperature-aware design and tools to support them.

Based on the previous and other similar thermal models, *dynamic thermal management (DTM)* techniques have been suggested for processors using architectural adaptation, *dynamic voltage scaling (DVS)*, *dynamic frequency scaling (DFS)* and profiling-based techniques. In Skadron et al. [2002], it is proposed to use formal feedback control theory as a way to implement adaptive techniques in the processor architecture. In Srinivasan and Adve [2003] a predictive frame-based DTM algorithm targeted at multimedia applications is presented. This algorithm uses profiling to predict the theoretical highest performance within a thermally safe HW configuration for the remaining frames of a certain type. Also, Brooks and Martonosi [2001] performed extensive studies on empirical DTM techniques (i.e., DVS, DFS, fetch toggling, throttling, and speculation control) when the power consumption of a processor crosses a predetermined threshold (i.e., 24W). Their results showed that both DFS and DVS can be very inefficient if their invocation time is not set appropriately. Additionally, Rohou and Smith [1999] suggested not scheduling hot tasks when the temperature reaches a critical level. In this way, the CPU spends more time in low-power states such that the temperature can be either locally or globally decreased. In this work we address the problem of empirical validating such approaches with long thermal simulations using real-life workloads, which becomes feasible with

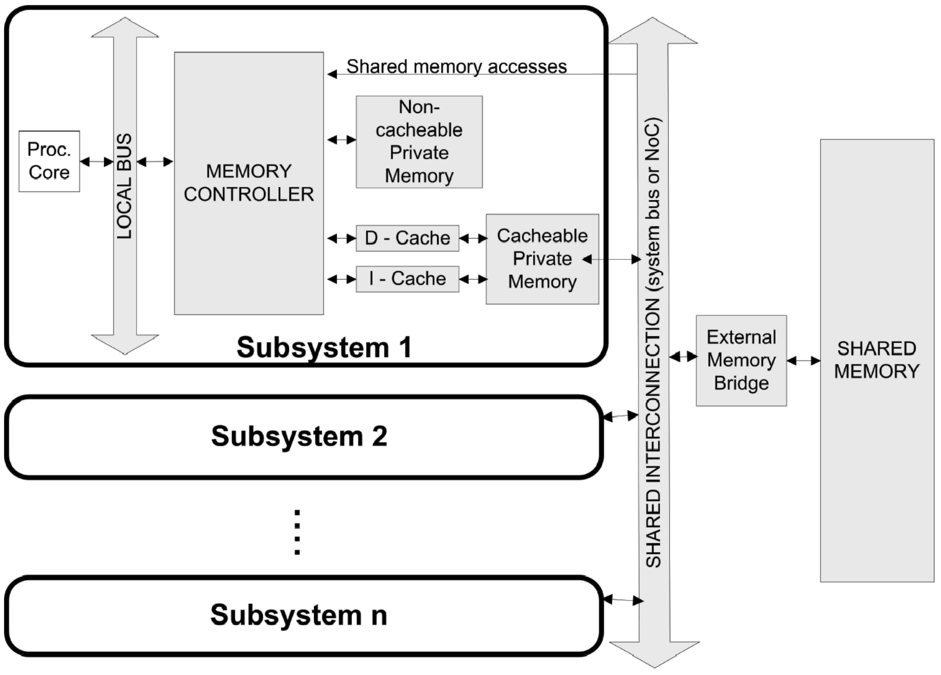

Fig. 1. Overview HW architecture of emulated MPSoCs.

our HW-SW thermal emulation tool. In fact, a simple DFS mechanism based on the previous works is presented in our experiments to illustrate the flexibility of the proposed HW-SW FPGA-based framework to interact with the SW part, and to explore in real time different temperature management policies.

Finally, another interesting line of research to ease the problem of temperature in future MPSoCs is temperature-aware placement [Chu and Wong 1998; Chen and Sapatnekar 2003; Goplen and Sapatnekar 2005]. In this case, the temperature issues are addressed at design time to ensure that circuit blocks are placed in such a way that they even out the thermal profile. All of these techniques are complementary to ours since we assume that the final floorplan and core placement phases have been already performed. Hence, our tool is able to take the outcome of any of the previous approaches and validate their predicted results during the execution of realistic applications of the target working environments.

3. MPSOC EMULATION ARCHITECTURE

The proposed MPSoC framework uses FPGA emulation as the key element to model the HW components of MPSoCs at multimegahertz speeds, and to extract the detailed system statistics used in our SW thermal library running in a host computer. An overview of the baseline HW architecture of the MPSoC emulation platform is depicted in Figure 1. It consists of three main elements:

(1) different MPSoC processing cores, such as Power PC, Microblaze, ARM, or VLIW;

(2) the definition of configurable I- and D-cache, as well as main memories (i.e., private and shared memories between processors); and
(3) various interconnection mechanisms between the L1 memory hierarchy and the main memory, namely, buses and NoCs.

These elements are designed in standard and parameterizable VHDL and mapped onto a Xilinx Virtex 2 Pro vp30 board (or V2VP30) with 3M gates, which costs approximately $2,000 in the market and includes two embedded Power PCs, various types of memories (i.e., SRAM and DDR), and an Ethernet port. However, any other FPGA could be used instead. The only requirements are the availability of an Ethernet core to interact with the SW thermal tool, a compiler for the included cores, and a method to upload both the FPGA synthesis of our framework and the compiled code of the application under study. In our case, Xilinx provides all these basic tools in its *embedded development kit (EDK)* framework for FPGAs.

In addition, note that the purpose of our emulator is not prototyping the final HW components in MPSoC systems, but constructing an emulation and fast exploration tool that can be used by designers to discover the desired characteristics and thermal effects of the eventual system. Therefore, our framework includes mechanisms to configure the exploration and hide those physical characteristics of the underlying HW that do not match the selected values (conversely to traditional prototyping), as will be explained further in Section 4.2.

In the following subsections we describe in detail the architecture and advanced emulation mechanisms of the different elements included in our emulation platform. We also depict synthesis figures for each component.

3.1 Processing Elements

In our framework, various types of processing cores can be included, both proprietary and public. The accepted input forms are netlists mapped onto the underlying FPGA and HDL languages (i.e., Verilog, VHDL, or synthesizable SystemC). This addition of cores is possible since the memory controller that receives the memory requests in our system includes an external pinout interface and protocol that can be easily modified to match the corresponding ones of the studied processor (Section 3.2). Moreover, only the instruction-set emulation part of the core is required because its memory hierarchy (e.g., caches or scratch pad) is replaced by our framework to explore different memory configurations.

In the current version of the system, we have ported a hard core (PowerPC 405) and a RISC-32 soft core (Microblaze) provided by Xilinx. None includes HDL sources, but only netlist mapping, and the inclusion process for their pinout interfaces and protocols required one week. Regarding platform scalability for MPSoC designs, a complete Microblaze requires only 4% of the total resources of our V2VP30 FPGA (574 out of 13,696 slices).

3.2 Memory Hierarchy

As Figure 1 indicates, in the basic emulated architecture two memory levels presently exist: L1 cache memories and main memories. However, it requires only a few minutes to add additional cache memory levels or private memories to each processing element, either on a per-processor or processor-group basis. The main element in the memory hierarchy that enables this easy integration of new memory devices and protocols is the memory controller. One memory controller is connected to each processing core to capture all its memory requests. Then, the memory controller forwards them to the necessary element of the memory subsystem according to the demanded memory address. In the current implementation, it takes 2% of the total available resources of our V2VP30 FPGA (270 slices), and includes interfaces and protocols for 4 memory components and 3 different memory address ranges:

(1) *private main memory, cacheable or noncacheable, addressable in a configurable memory range of each processor*: It is possible to configure its size and latency, so long as enough *block random access memory (BRAM)* resources exist. Its synthesis takes 1% of the V2VP30 (181 slices), apart from the used BRAM that depends on the desired size.
(2) *shared main memory, cacheable or noncacheable according to the user's configuration*: It is possible to configure its total size and latency, and it does not take any area in the FPGA since it uses real memories available on the board, namely, *synchronous random access memories (SRAM)* or *double data rate synchronous dynamic RAM (DDR-SDRAM or DDR memories)*.
(3) *private HW-controlled D- and I-caches*: It is possible to define independently the total D- and I-cache sizes, line sizes, and latencies to explore different design alternatives. In our experiments, both caches are directly mapped. However, their modular designs include in different concurrent processes the replacement policy and associativity features, making it easy to change this configuration with additional algorithms to test. Its synthesis uses 1% of the V2VP30 (181 slices) and the amount of used BRAM varies according to the desired size.

Finally, each memory controller is able to observe and synchronize different clock domains due to its multiple external interfaces (see Figure 1). It has internal counters for each type of connected memory to keep track of the elapsed time and compare it with user-defined latencies. Then, the memory controller informs the *virtual platform clock manager (VPCM)* (see Section 4.2) to stop the processor's clock during the emulation each time one physical memory device cannot fulfill the defined latency. Hence, the stopped processor preserves its current internal state until it is resumed by the clock manager, when the memory controller informs it that the information requested is available. This mechanism enables tradeoffs between emulation performance and use of resources, as detailed in Section 4.2. Currently, our memory controller monitors two clock domains: one for the microprocessor and the other for the memories and memory controller itself.

3.3 Interconnection Mechanisms

The third configurable element in our MPSoC emulation framework is the interconnection mechanism between the memory controller and main memory (i.e., in BRAM, SRAM, or DDR memories). At this level, we have included both buses and NoCs. To enable this variety of choices, apart from multiple types of interfaces of the memory controller, we have also included a configurable main memory bridge in the device side. It includes two different public pinout interfaces: one corresponds to the memory and the other to the instantiated interconnection. Similarly as the memory controller, this enables us to extend the current list of available interconnection mechanisms by modifying the required pinout and protocol.

In the current version, two available buses on Xilinx FPGAs are included, namely, an *on-chip peripheral bus (OPB)* for general-purpose devices and a *processor local bus (PLB)* for fast memories and processors. Also, we have created our own 32-bit data/address bus for exploration purposes. It is inspired by the basic functionality of the AMBA 2 AHB interconnect [ARM 2003], where the bandwidth and arbitration policies can be configured. Thus, starting from the initial OPB scheme, we have removed the signals used for advanced arbitration schemes, transaction parking request, etc. Also, the arbitration protocol is specified at compile time, hence avoiding the need for a dedicated signal to set the arbitration mode. Currently, the allowed arbitration modes are: priority-based and round robin. In addition, the latency and bus width can be configured. For our experiments (see Section 7) the arbitration latency is one cycle, and connected to all processing cores through an OPB interface and to an external SRAM memory through a custom SRAM controller. Its synthesis (including the SRAM controller) represents 1% of the V2P30 FPGA (210 slices).

In addition, we have included a facility for exploring custom-made NoC solutions. The synthesizable NoC code is generated using the Xpipes NoC compiler [Jalabert et al. 2004]. It allows for studying topologies with any number of switches, links with bandwidth constraints, and NIs to connect external cores to the NoC. We have modified the memory controller and main memory bridges to generate *open core protocol (OCP)* transactions as the Xpipes NIs require [Jalabert et al. 2004]. Regarding FPGA utilization, a complex NoC-based system with 6 switches of 4 input/output channels and 3 output buffers uses 70% of the V2P30 FPGA (9,659 slices).

The inclusion of each of these buses and NoC interfaces in our framework required one week of work. Furthermore, as with processing cores, any other high-performance proprietary bus (e.g., AMBA, STBus, etc.) can be added to our emulation framework as a black box, since the integration process only requires to know the used protocol and external bus pinout.

4. STATISTICS EXTRACTION SUBSYSTEM

The main feature pursued in the design of the statistic extraction subsystem is its transparent inclusion in the basic MPSoC architecture to be evaluated, and with minimum performance penalty in the overall emulation process. For this purpose, as depicted in Figure 2, we have implemented HW sniffers (see Section 4.1) that monitor certain signals of the memory controller and the external pinout of each device included in the emulated MPSoC. These sniffers calculate, among other statistics, the energy consumed in each cell (see Section 5)

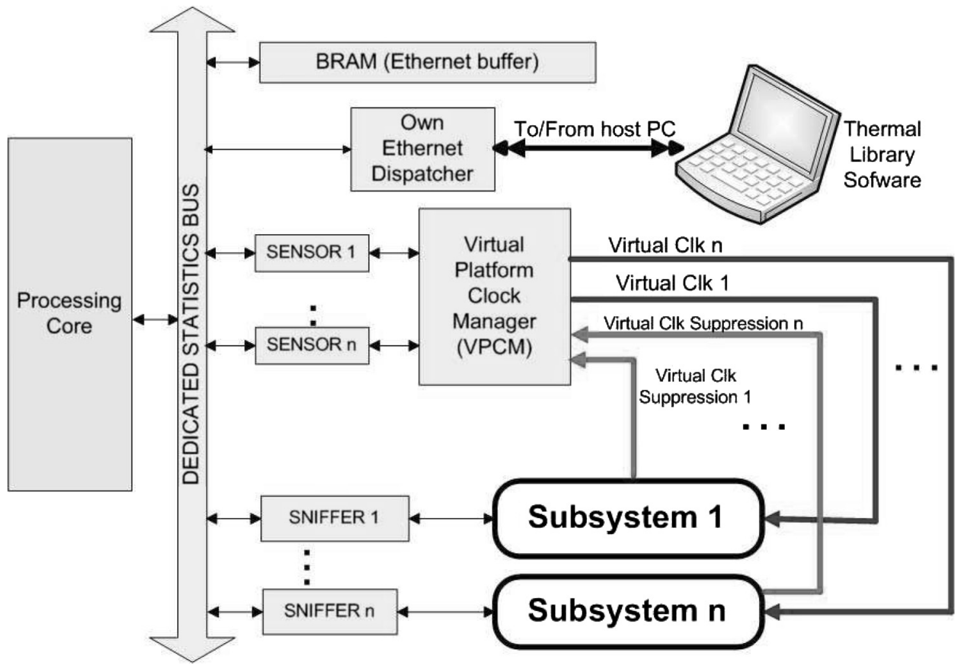

Fig. 2. Overview of the statistics extraction subsystem.

of the floorplan of the emulated MPSoC, and store the final values in a buffer created in the FPGA BRAM memory. Finally, the buffers are concurrently processed by our network dispatcher to generate *medium access control (MAC)* packets in our own format, and sent by an Ethernet port to the SW thermal modeling library running in the connected computer. One key additional element in this extraction mechanism is the VPCM module (see Section 4.2), which enables stopping and resuming the statistics extraction mechanism in the case of congestion of the Ethernet connection.

4.1 HW Sniffers

HW sniffers transparently extract the statistics from each MPSoC component defined in the floorplan. From a design point-of-view, all sniffers in our platform share a common structure. They have a dedicated interface to capture internal signals from the module they are monitoring, and a connection to our custom statistics bus. To create a new sniffer, the designer need only define what to monitor in the component and how to connect the sniffer to the bus. For temperature monitoring, HW sniffers measure the time that each processor spends in active/stalled/idle mode at runtime, and the number and types of access to memories in the system (i.e., I- and D-caches, and large shared and private main memories). At the interconnection level (buses or NoC), the monitored values are the number of signal transitions. There is an skeleton available to ease the creation of new sniffers for this purpose. Currently, we provide two different types of sniffers. The first, called event logging, exhaustively logs all interesting events that occur in the platform. The second, called count logging, only counts events such as cache misses, bus transactions, memory accesses, etc.; thus, it generates more concise results, which is typically what designers demand from cycle-accurate simulators to test their systems. Our experimental results with real-life MPSoC designs (see Section 7) indicate that practically an unlimited number of event-counting sniffers can be added to the design without deteriorating at all the emulation speed. This establishes one of the main differences from SW cycle-accurate simulation systems: The addition of additional cores or analysis sniffers to the MPSoC architecture does not slow down the emulation process due to the implicit concurrent synchronization of signals between different HW modules working in parallel to compose a complete MPSoC architecture. In fact, HW sniffers merely act as an additional HW component that transparently monitors the switching activity of signal and internal states of the bare MPSoC HW architecture.

Finally, as an example to evaluate how much FPGA area overhead the statistics extraction subsystem represents, the amount of resources used by one event-logging sniffer is 0.1% (14 slices), while for an event-counting sniffer it is about 0.2% (31 slices).

4.2 Virtual Platform Clock Manager (VPCM)

The VPCM is the HW element used in our framework to provide multiple virtual clock domains. This module generates as output the clock signals used in the emulated MPSoC subsystems (VIRTUAL CLK signals in Figure 2). It receives three different types of input signals. First, the physical clock is generated in the oscillator of the FPGA (not shown in Figure 2 for simplification purposes), which in the current implementation is set to 100 MHz. Second, one signal from each memory controller of the emulated MPSoC subsystems (VIRTUAL CLK SUPPRESSION 1..N in Figure 2) is used to request a virtual clock inhibition period if any attached memory device of the emulated hierarchy is not able to return the requested value at this moment, respecting its set user-defined latency (see Section 3.2). Third, signals coming from different virtual temperature sensors (SENSOR 1..N in Figure 2) monitor whether any component has increased its temperature beyond/below a certain threshold. This mechanism enables the use of runtime thermal management policies (see Section 7 for examples). Virtual temperature sensors are regular registers that store the current updated runtime temperature coming from the SW thermal library running in the host computer. However, in the final MPSoC they would be replaced by real sensors. Then, the use of virtual clock domains generated by the VPCM is twofold:

—First, the emulation of MPSoCs can be done for physical features other than those of available HW components. Once the respective VIRTUAL CLK SUPPRESSION 1..N signal is high, the corresponding VIRTUAL CLK signal of that subsystem (or set of subsystems) is activated. Then, the stopped processor preserves its current internal state until it is resumed by the VPCM, when the memory controller states that the information requested is available in the accessed memory. This mechanism allows us to implement the corresponding memory resources either in internal FPGA memory (for optimal performance) or with external memories (for bigger size), while balancing emulation performance and the use of resources. For instance, if the desired latency of main memories is 10 cycles but the available type of memory modules in the FPGA is slower (e.g., use of DDR instead of SRAMs), the VPCM can stop the clock of the processors involved at runtime. Thus, it can hide the additional clock cycles required by the memory. Our VPCM includes two clock domains: (1) microprocessor, memories, and interconnections; and (2) memory controllers.

—Second, the virtual clock of all or part of the components in the emulated MPSoC can be transparently stopped/resumed at runtime in the case of saturation of the Ethernet connection during download/upload of the extracted statistics/estimated temperatures.

The combination of these two mechanisms enables the execution and thermal modeling of HW configurations of the emulated MPSoC at a different speed from the allowed clock speed of available HW components. In fact, it is similar to the mechanism used in SW simulations, but at a much higher frequency (see Section 7). For instance, it is possible to explore the thermal modeling effects of a final system clocked at 500 MHz, even if the present cores of the FPGA can only work at 100 MHz. To this end, instead of using a 10 ms statistics sampling frequency with a desired virtual clock emulation of 500 MHz, our framework uses a virtual clock of 100 MHz (the maximum clock allowed in FPGA emulation after synthesis). This clock is $5\times$ slower than the desired emulated clock and collects the statistics every 50 ms, but the switching activity in each MPSoC component monitored at this interval is equivalent to target system for 10 ms. Therefore, our framework samples every 50 ms of real execution, but is analyzed by the SW thermal library as representing 10 ms of the target MPSoC emulated execution. The major requirement in this case is the definition of the sampled/emulating frequency and the target MPSoC frequency to configure the SW thermal model accordingly. The SW thermal model is described next.

5. MPSOC SW POWER/THERMAL MODELING

Our SW thermal tool is a C++ library that enables thermal exploration of silicon bulk chip systems. It can evaluate the thermal behavior in devices modeled at different levels of abstraction (i.e., gate-level, RTL-level, and architectural-level). The switching activities of the wires and components in the die for this thermal analysis are obtained from our FPGA-based MPSoC emulation (see Section 3). Then, the library can be configured in multiple ways to evaluate the thermal behavior of different alternatives for each final MPSoC chip. For instance, its space resolution for thermal accuracy is configurable (i.e., number of temperature cells in a fixed area), as are many other packaging parameters (e.g., quality of heat sink, thermal capacitance of the different materials that compose the chip, etc.). In our experiments (Section 7), this flexibility in thermal library configuration is used to investigate the runtime thermal behavior of multiple cores and embedded memories on a single die in the case of different package solutions and floorplan designs for MPSoCs.

Table I. Power for the Most Important Components of an MPSoC Design (using a 0.13 μm bulk CMOS technology)

MPSoC Component	Max. Power (at 100 MHz)	Max. Power (at 500 MHz)	Max. Power Density
RISC 32-ARM7	5.5mW	—	$0.03W/mm^2$
RISC 32-ARM11	0.3W	1.5W	$0.52W/mm^2$
DCache 8kB/DM (ARM7)	28mW	—	$0.028W/mm^2$
DCache 8kB/DM (ARM11)	142mW	710mW	$1.97W/mm^2$
ICache 8kB/DM (ARM7)	28mW	—	$0.028W/mm^2$
ICache 8kB/DM (ARM11)	142mW	710mW	$1.97W/mm^2$
Memory 32kB (ARM7)	11mW	—	$0.01W/mm^2$
Memory 32kB (ARM11)	55mW	275mW	$0.76W/mm^2$
NoC switch (6x6-32b)	56mW	257mW	$0.08W/mm^2$
NoC network interface	23mW	128mW	$0.02W/mm^2$

In the next subsections, we first discuss the utilized power model. Second, we explain the thermal model in detail. Finally, we review the thermal calculation speed and accuracy of the current library implementation.

5.1 Power Estimation

In Table I, we outline the power consumption and power densities for the most important components of the evaluated MPSoCs as an illustration of what our tool requires. We use, as Table I indicates, the maximum power numbers for each component as the worst case, but the effective power can normally be lower, depending on the workload (activities of processors and memories), and be given as an input by the designer for his particular design. These values have been derived from industrial power models for a 0.13 μm technology.

Regarding leakage power, we currently assign a fixed 10% weight to leakage energy. This figure actually corresponds to the indications of the *international technology roadmap for semiconductors (ITRS)* [SIA 2004] for low-standby power systems in 0.13 μ with a supply voltage of 1.2–1.3V. ITRS outlines that in this case Vdd/Vt are very aggresively scaled to guarantee sufficient battery lifetime. Hence, using an optimal Vdd/Vt operating point results in very limited leakage power variation for different working temperatures. However, in more recent technology nodes, leakage variations become more important, and our SW thermal library and HW sniffers can be extended accordingly.

5.2 Thermal Estimation

In our case we consider MPSoCs HW that is made of silicon die wrapped into a package placed on a *printed circuit board (PCB)*, with variable cost (from low-cost to high-cost packaging, as shown in Section 7). In this case, as shown in Figure 3, the heat flow starts from the bottom surface of the die and goes up to the silicon, passes through the heat spreader, and ends at the environment interface where the heat is spread by natural convection [Skadron et al. 2004]. Therefore, for modeling the heat flow, we rely on an equivalent electrical RC model (see Figure 3). Two RC models are supported. On the one hand, we have developed our own thermal model of MPSoC considering nonlinear resistances

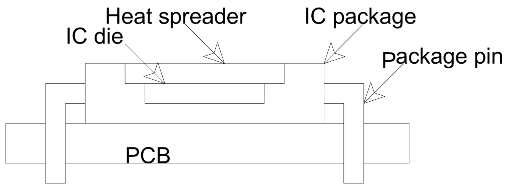

Fig. 3. Chip packaging solution.

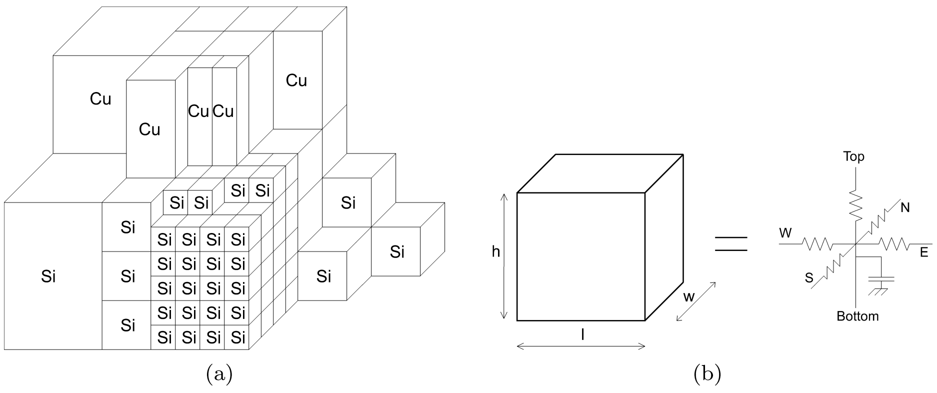

Fig. 4. (a) Chip divided into cells; (b) equivalent RC circuit.

inside the silicon [Paci et al. 2006] in order to match the behavior of thermal conductivity. We consider the heat spreader made of copper and use linear resistances to model it. Currently, we can analyze 2 seconds of simulation (in a 660-cell floorplan) in 1.65 seconds on a P-4 at 3 GHz, which is fast enough to interact in real time with our FPGA-based MPSoC emulation. On the other hand, we have cross-checked our results by including in our tool the capability to use the Hotspot v3.0 thermal model [Skadron et al. 2004] (see Section 7 for more details). It is an accurate model for high-performance processors, based on an equivalent circuit of linear thermal resistances and capacitances which correspond to microarchitecture blocks and essential aspects of the thermal package. This model has been validated using finite element simulation. In the following subsections, we further describe our own thermal model and refer to Skadron et al. [2004] for a more detailed explanation of this library.

5.2.1 *Modeling the Heat Flow.* A low-power MPSoC is usually packed within a plastic ball grid array package [Vandevelde et al. 2001] (see Figure 3). In our library, we assume that all surfaces but that of the heat spreader are adiabatic. The spreader disposes the generated heat by natural convection with the ambient.

Then, similar to Skadron et al. [2004], Su et al. [2003], and Heo et al. [2003], we exploit the well-known analogy between electrical circuits and thermal models. We decompose the silicon die and heat spreader in elementary cells which have a cubic shape (see Figure 4) and use an equivalent RC model for computing the temperature of each cell. By varying the cell size and number of

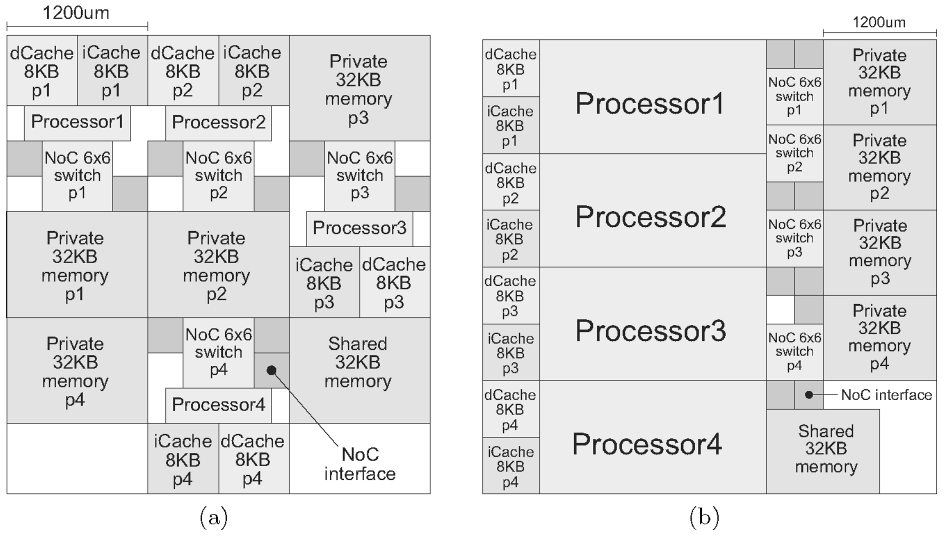

Fig. 5. MPSoC floorplan with (a) 4 ARM7 cores; and (b) 4 ARM11 cores.

cells, we can tradeoff the simulation speed of the thermal library with its accuracy. In our experiments we have used two basic floorplans: (a) 4 ARM7 cores at 100 MHz; and (b) 4 ARM11 running at 100 or 500 MHz, both in 0.13 μm technology (see Figure 5). The interconnect is clocked at the same frequency as the cores in each case. The cell sizes used in both cases are $150um * 150um$. We assume that the power is uniformly burned in this region, which represents 1/8th of the size of an ARM processor in 0.13 μm. For technologies with worse thermal conductance, such as the fully depleted silicon-on-insulator [SIA 2004], it is possible to use smaller thermal cells (down to the level of standard cells).

5.2.2 *Equivalent RC Thermal Model.* We associate with each cell a thermal capacitance and five thermal resistances (see Figure 4). Four resistances are used for modeling the horizontal thermal spreading, whereas the fifth is used for vertical thermal behavior. The thermal conductivity and capacitance of each cell is computed as follows (where $k_{th}^{si/cu}$ is the thermal conductivity and $c_{th}^{si/cu}$ is the thermal capacitance per unit volume):

$$G_{th}^{NESW} = k_{th}^{si/cu} \cdot \frac{h \cdot w}{l} \tag{1}$$

$$G_{th}^{top} = k_{th}^{si/cu} \cdot \frac{l \cdot w}{h} \tag{2}$$

$$C_{th} = c_{th} \cdot l \cdot h \cdot w \tag{3}$$

We model the generated heat by adding an equivalent current source to the cells on the bottom surface. The heat injected by the current source into the cell corresponds to the power density of the architectural component covering the cell (e.g., a memory decoder or processor) multiplied by the surface area of the cell. No heat is transferred down into the package from these bottom cells.

Table II. Thermal Properties

silicon thermal conductivity	$150 \cdot \left(\frac{300}{T}\right)^{4/3} W/mK$
silicon specific heat	$1.628e - 12J/um^3K$
silicon thickness	$350um$
copper thermal conductivity	$400W/mK$
copper specific heat	$3.55e - 12J/um^3K$
copper thickness	$1000um$
package-to-air conductivity (low-cost)	$40K/W$

In contrast, the heat from cells on the top surface is removed through convection. We model this by connecting an extra resistance in series where $R_{th}^{top} = 1/G_{th}^{top}$ resistance. The value of this resistance is equal to the package-to-air resistance weighted with the relative area of the cell to the area of the spreader.

Finally, the thermal model is calibrated against a 3D finite element analysis given by an industrial partner.

5.2.3 *Thermal Properties.* In Table II we enumerate thermal properties of the different packaging options used during our experiments. The amount of heat that can be removed by natural convection strongly depends on the environment, such as the placement of the chip on the PCB as in the case of embedded systems. Regarding package-to-air resistance, we consider the case of very low-cost packaging where a good average value is 42W/K (see Vandevelde et al. [2001]) because of the uncertainty of the final MPSoC working conditions. However, since this value is higher than the actual figures published by some package vendors, in our experiments (see Section 7) we also study the effect of different packaging solutions for MPSoCs.

6. HW-SW MPSOC EMULATION FLOWS

The key advantage of our framework for exploration of MPSoC designs with thermal management at high speed is its double integration of statistics extraction (from HW emulation) and SW thermal simulation of all MPSoC architectural blocks in one overall tool flow. The whole system flow is depicted in Figure 6.

First, the HW and SW components of the system are defined. Regarding HW, the user specifies in this phase one concrete architecture and all the HW sniffers. These HW sniffers are to extract statistics for each of the three main architectural levels that constitute the final MPSoC: processing cores, memory subsystem, and interconnection to the main memories. This is done by instantiating, in a plug-and-play fashion, the predefined HDL modules available in our repository for each of the aforementioned three levels (see Section 3) and corresponding sniffers. In our case we use the Xilinx integrated software environment. As for the SW part, in this phase it is compiled in the application(s) to be tested in the emulated MPSoC. In our case we use Xilinx EDK, which includes GNU C (gcc) and C++ (g++) compilers/linkers for the Power PC and the Microblaze cores available in our repository. Also, EDK enables

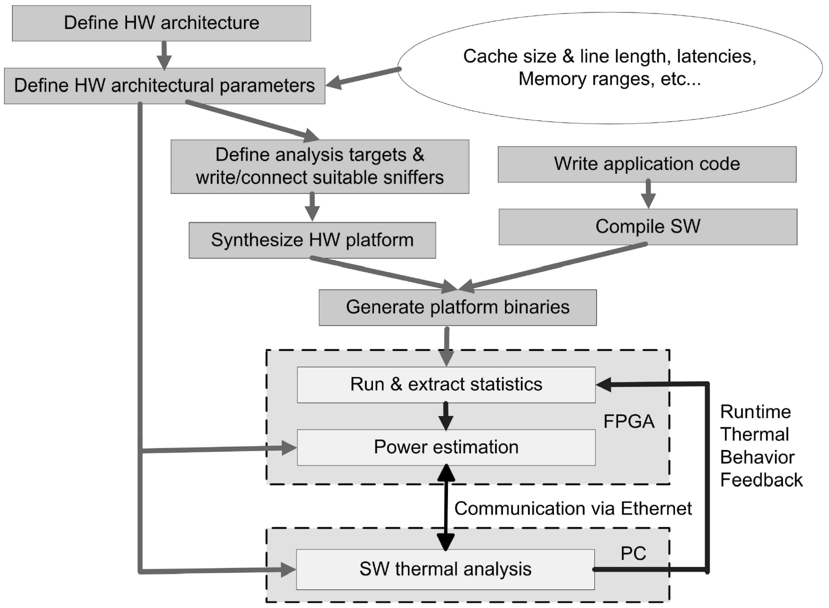
Define HW architecture
Define HW architectural parameters
Cache size & line length, latencies, Memory ranges, etc...
Define analysis targets & write/connect suitable sniffers
Write application code
Synthesize HW platform
Compile SW
Generate platform binaries
Run & extract statistics
FPGA
Power estimation
Runtime Thermal Behavior Feedback
Communication via Ethernet
PC
SW thermal analysis

Fig. 6. Complete HW-SW flows included in the FPGA-based thermal emulation framework.

loading different binaries on each system processor. Thus, if the application to be tested is already written in any of these languages, no effort is required for the designer, since the memory hierarchy and utilization of the interconnection mechanism (e.g., generation of OCP transaction for the NIs of the NoC) are transparently generated by the underlying emulated HW architecture. For a complex MPSoC with 8 processors and 20 additional HW modules, this phase requires 10 to 12 hours overall, including the complete synthesis phase with standard tools. Moreover, modifications in the current core configurations take less than 1 hour to resynthesize, while the compilation of the additional SW part of a 4-processor emulation system only takes minutes.

In the next phase, the floorplan to be evaluated according to the previous HW definition is specified. At this time the different energy and frequency values for each HW component in the emulated MPSoC are set. Also, the configurable granularity of the temperature updates and communication between the FPGA and SW thermal library is configured. This value is fixed at 10 *ms* in our experiments (Section 7).

Next, the whole HW emulated MPSoC is uploaded onto the Xilinx FPGA-based platform using a JTAG device, and the graphical interface of our SW thermal model is launched in the host computer. After this point our framework runs autonomously. While the emulated system is running, the statistics concerning the power values for each cell defined in the layout are concurrently extracted, and sent to the thermal simulator running onto the host computer via a standard Ethernet connection. The thermal simulator calculates in real time the new temperatures and feeds back these updates by sending MAC packets to the FPGA-based emulation framework. According to this newly received

Table III. Timing Comparisons between Our MPSoC Emulation Framework and MPARM

	MPARM	HW Emulator
Matrix (1 core)	106 sec	1.2 sec (88×)
Matrix (4 cores)	5 min 23 sec	1.2 sec (269×)
Matrix (8 cores)	13 min 17 sec	1.2 sec (664×)
Dithering (4 cores-bus)	2 min 35 sec	0.18 sec (861×)
Dithering (4 cores-NoC)	3 min 15 sec	0.17 sec (1147×)
Matrix-TM (4 cores-NoC)	2 days	5' 02 sec (1612×)

information, the implemented temperature manager in our FPGA can be used to test different runtime thermal management policies on the emulated MPSoC (see Section 7 for an example).

7. EXPERIMENTAL RESULTS

We have assessed the performance and flexibility of the proposed emulation framework in comparison with the MPARM framework [Benini et al. 2005] and its internal SW thermal library by running several examples of multimedia and intensive processing cores of MPSoC designs (see Sections 7.1 and 7.2). Additionally, our experiments include applying the presented framework to test a runtime DFS mechanism for one complex MPSoC case study based on ARM-11 cores (see Section 7.3), with various packaging techniques (Section 7.4), and different thermal-aware floorplan solutions (Section 7.5). In our experiments MPARM is executed on a P-4 at 3.0 GHz with 1GB of SDRAM and running GNU/Linux 2.6.

7.1 MPSoC Emulation vs. Simulation Performance Evaluation

In the first set of experiments we have assessed the performance of the bare MPSoC emulation framework (without thermal modeling) for system architecture exploration, in comparison to cycle-accurate simulators. To this end, we evaluated various configurations of interconnections and processors (1 to 8) using a complex L1 hierarchy for each core with a 4KB D-cache/I-cache, 16KB of private memory, and a global 1MB main shared memory. All processors used OPB and OCP buses. As an example, the MPSoC design with HW sniffers and 4 processors (1 hard-core PowerPC and 3 soft-core Microblazes) consumes 66% of the V2VP30 and runs at 100 MHz. Next, we explored the use of NoCs [Jalabert et al. 2004] instead of buses. The tested NoC had two 32-bit switches with 6 inputs/outputs and 3-package buffers. This NoC-based MPSoC required 80% of our FPGA.

For the SW drivers, first we used a kernel application (MATRIX in Table III) that performs independent matrix multiplications at each processor private memory and combines the results in memory at the end. Second, we used a dithering filter (DITHERING in Table III) using the Floyd algorithm [Floyd 1985] in two 128x128 grey images, divided into 4 segments and stored in shared memories. This application is highly parallel and imposes almost the same workload in each processor. The obtained timing results are depicted in Table III.

These results show that the HW-SW emulation framework scales significantly better than SW simulation. In fact, an exploration of MPSoC solutions with 8 cores for the Matrix driver took 1.2 seconds per run in our case, but more than 13 minutes in MPARM (at 125 KHz), resulting in a speedup of 664×. Moreover, the exploration of NoCs with complex SW drivers (dithering with 4 cores, 30 HW MPSoC components in total) shows larger speedups (1147×) due to signal management overhead in cycle-accurate simulators (Table III). As a result, our HW-SW emulation framework achieved an overall speedup of more than three orders of magnitude (1147×), illustrating its clear benefits for exploration of the design space of complex MPSoC architectures compared to cycle-accurate simulators.

7.2 MPSoC Thermal Modeling Using Cycle-Accurate Simulation vs. HW-SW Emulation

In the second set of experiments we have verified the capabilities of real time interaction between the HW FPGA-based emulation and SW thermal library components of our system, compared to pure cycle-accurate SW simulation. In this case we considered a low-cost package solution (see Table II). From the HW viewpoint, we defined a system with 4 RISC-32 processing cores. Each core was attached to a local 8KB direct-mapped instruction and data caches, using a write-through replacement policy. Also, each processor had a 32KB cacheable private memory, and a 32KB shared memory was included in the system. The memories and processors were connected using an XPipes NoC of four 6x6 switches and NI modules. The considered floorplan is shown in Figure 5 and included 128 thermal cells. We obtained the dimensions of the NoC circuits by synthesizing and building a layout. As the SW driver for this MPSoC design, we defined a benchmark (Matrix-TM in Table III) that keeps the processor workload close to 100% all of the time, pushing the MPSoC to its processing power limits to observe effects in temperature. This benchmark implements a pipeline of 100K matrix multiplication kernels based on the Matrix benchmark (see Table III). Each processor executes a matrix multiplication between an input matrix and a private operand matrix, then feeds its output to the logically following processor. The platform receives a continuous flow of input matrices and produces a continuous flow of output matrices. Every core follows a fixed execution pattern: (i) copy of an input matrix from the shared memory to its private memory; and (ii) multiplication of the new matrix with a matrix already stored in the private memory; (iii) copy of the resulting matrix back to the shared memory. During the whole execution, interrupt and/or semaphore slaves are queried to keep the synchronization, creating an important amount of traffic to the memories. The obtained timing results (Table III) show that our HW-SW emulation framework takes approximately 5 minutes for the entire execution of the driver, including thermal monitoring, versus 2 days in MPARM for just 0.18 seconds of real execution (left corner in Figure 7); Thus, our framework achieves more than three orders of magnitude in speedups (1612×) compared to SW-based thermal simulation, making feasible to study, in a reasonable time, long thermal effects.

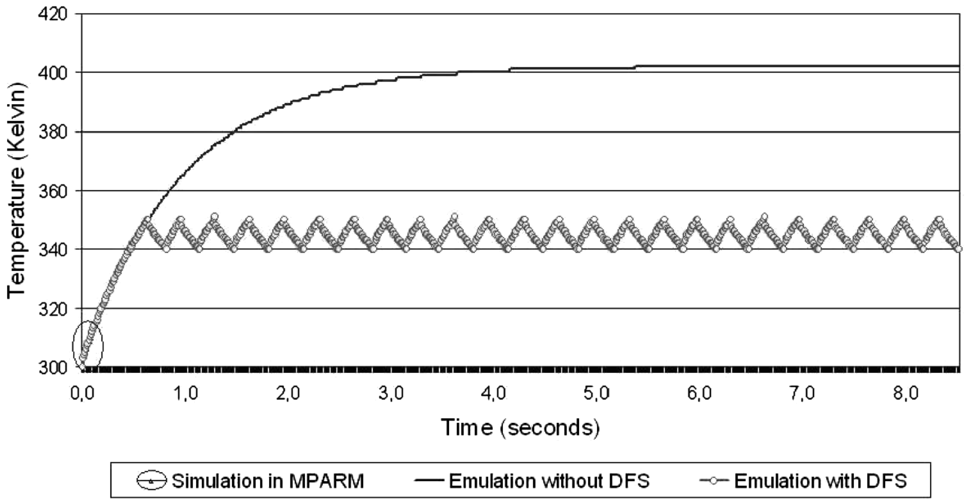

Fig. 7. Average temperature evolution of Matrix-TM in a 4-core MPSoC at 500 MHz or using a two-choice DFS (500–100MHz).

7.3 Evaluation of Dynamic Thermal Strategies in MPSoCs

In the third set of experiments we have performed a long thermal emulation in our framework to observe thermal effects on the MPSoC with real-life processing inputs of embedded applications. We ran the Matrix-TM workload for 100K iterations and the results for a 500 MHz emulation are shown in Figure 7. These indicate the need to perform long emulations to estimate thermal effects (note in Figure 7 that the previous simulation in MPARM only represents a very limited part of the overall MPSoC thermal behavior). Due to the high rise in temperature observed in the MPSoC design, we explored the possible benefits of DTM techniques within our HW-SW emulation framework. To this end, we implemented a simple threshold monitoring policy using the available HW temperature sensors in our framework. The policy consists of a simple dual-state machine that monitors at runtime whether the temperature of each MPSoC component increases/decreases above/below two certain thresholds that we defined (350 or 340 degrees Kelvin in this example). Then, the temperature sensors inform the VPCM, which performs DFS choosing between 500 or 100 MHz accordingly. The results are also shown in Figure 7 and indicate that this simple thermal management policy could be highly beneficial in MPSoC designs using low-cost packaging solutions (i.e., with values of package-to-air resistance of more than 40K/W). Furthermore, these results outline the potential benefits of our HW-SW emulation tool for exploring the design space of complex thermal management policies in MPSoCs, compared to SW cycle-accurate simulators that suffer from important speed limits.

7.4 Floorplan Selection Exploration in MPSoCs

When an integrated system is built for a certain MPSoC, the definition of an appropriate floorplan is a very complex task for system integration designers.

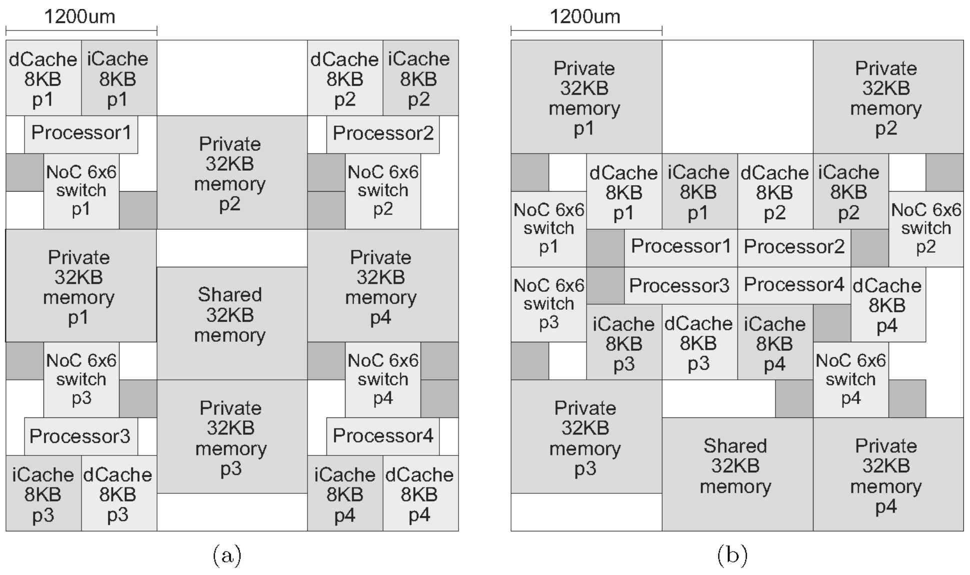

Fig. 8. MPSoC floorplan with cores; (a) scattered in the corners; and (b) clustered in the center of the chip.

In fact, deciding a suitable placement for each block in the MPSoC architecture requires taking into account multiple constraints (e.g., power, energy, performance, etc.) with values that are specific for each design. Recently, due to the increasing temperature in MPSoCs, thermal behavior has become another key factor defining the placement of each block of the design [Chu and Wong 1998; Chen and Sapatnekar 2003]. In this set of experiments we have used our tool to evaluate two additional thermal-aware floorplans (see Figure 8) for our initial case study with four processing cores and a NoC-based interconnect working at 500 MHz (see Figure 5). The first alternative floorplan scatters the processing cores in the corners of the chip (Figure 8(a)), while in the second all the cores are clustered together in the center (Figure 8(b)). We assumed the use of a low-cost packaging solution in all cases (see Table II).

The results are shown in Figure 9. In this case we can observe that the best floorplan to minimize temperature (15% less heating speed on average than the initial floorplan of Figure 5) was achieved with the placement technique that tries to assign processing cores to the corners of the layout (labeled as `scattered` in Figure 9). Hence, this solution is the best of the three thermal-aware placement options because it most delays the need to apply the available DFS mechanism shown in Figure 9, although its interconnects experience more heating effects due to the longer and more conflicting connection paths between components, which can originate more NoC congestion effects. By contrast, the solution that tries to place all processing cores in the center of the chip (labeled as `clustered` in Figure 9) shows the worse thermal behavior, but just slightly worst in temperature (5% on average) than the original manual placement of cores used for this MPSoC design, while the delays in interconnections between cores are minimal for the former due to their closest locations in the floorplan

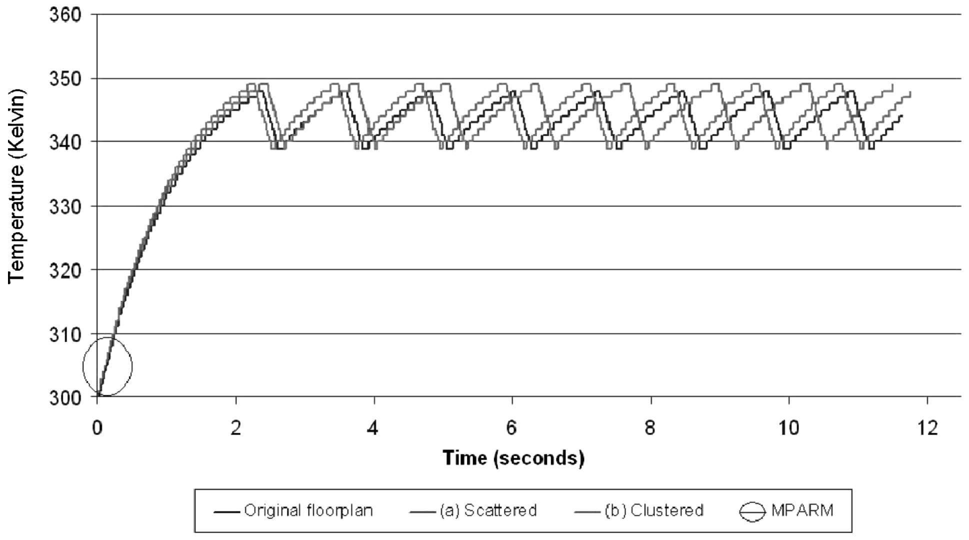

Fig. 9. Average temperature evolution with different floorplans for Matrix-TM at 500 MHz with DFS on.

(see Figure 8(b)). The main conclusion from this study is that a more aggresive temperature-aware placement must be applied (e.g., placement of cores scattered in the corners of the chip) to justify placing the cores apart, as attempted in the original manual design, to compensate for the heating effects on the chip due to longer interconnects. Otherwise, the possible penalty for long interconnects may not be justified, since a uniform distribution of power sources does not necessarily lead to a uniform temperature in the die. Moreover, these results clearly outline the importance for tool designers to explore the concrete thermal behavior of each design, and to select the most appropriate placement in an early stage of integration flow.

7.5 Effect of Different Packaging Technologies and SW Thermal Libraries

In this final set of experiments we tested different packaging solutions and compared them with the thermal behavior of the low-cost value of 40K/W that was initially considered (Table II) for our MPSoC floorplan with four RISC-32 processing cores working at 500 MHz with a NoC interconnect (see Figure 5). We simulated this floorplan with two additional values, namely, 12K/W in the case of standard packaging [ARM 2004b] and 5K/W in the case of high-cost and high-performance embedded processors [AMD 2004]. The results are shown in Figure 10.

As this figure shows, in the case of the standard packaging solution, the MPSoC design required more time to heat up and reached a maximum value of 360 degrees Kelvin when the DFS mechanism was not applied. This is lower than the case of low-cost packaging (40K/W), which reached a temperature of more than 500 degrees Kelvin. However, the thermal behavior of the standard packaging system was similar to the low-cost solution (its starting point was only slightly shifted to the right due to its more gradual temperature-rise

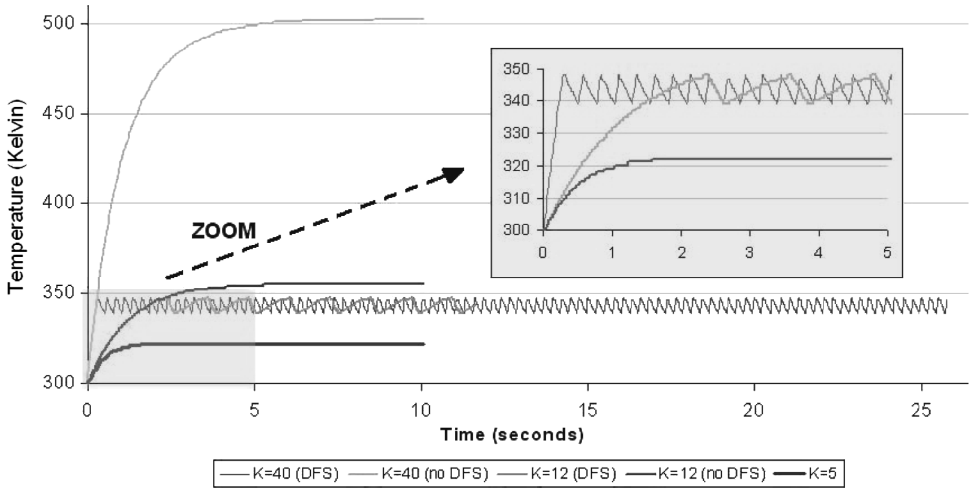

Fig. 10. Thermal behavior for an MPSoC floorplan using low-cost, standard, and high-cost packaging solutions.

curve) when the presented threshold-based DTM strategy of Section 7.3, fixed at 250 degrees Kelvin, was applied. Therefore, with this threshold value, no significant improvements were obtained with the standard package, and the low-cost solution would be preferable for this design using DTM. However, for the high-cost packaging solution (for 5K/W), the system showed a completely different temperature behavior where the chip never went beyond 325 degrees Kelvin. Therefore, this packaging solution creates much less thermal stress in the overall MPSoC implementation, and does not require the application of DFS because the design never reaches a temperature above the 350-degree-Kelvin threshold. As a result, this solution could significantly increase the expected mean-time-to-failure of the component and be interesting in highly reliable versions of this MPSoC chip design. However, note that this type of package has the important drawback of high cost for the manufacturer of the final embedded system, namely, typically 5 to 12× more than standard package solutions and more than 20× the cost of the low-cost package solution [IBM 2006]; Thus, it can seriously increase the price of the final product and developers would like to avoid it if possible. Hence, this type of experiment and presented framework can be a very powerful tool for designers to decide which type of packaging technique would be enough for a specific set of constraints in forthcoming generations of MPSoC designs.

Finally, we performed the same set of emulation experiments replacing our library with the well-known Hotspot v3.0 thermal library [Skadron et al. 2004], configuring it with the same packaging options previously tested. The results of this additional set of experiments show a very similar thermal behavior in comparison to our own library in the case of high-cost packaging (less than 3 degrees Kelvin of difference), which is the original target of the Hotspot library. In cases, of low-cost and standard packaging, variations that range between 4 to 15 degrees Kelvin have been observed. The origin of these variations the nonlinear dependency factor of silicon thermal conductivity with respect to the actual temperature in the die, which is included in our own library, but not modeled in the Hotspot library. In fact, our results indicate that this nonlinear part of the thermal equations is particularly important when the temperature rises beyond 360 degrees Kelvin in the case of low-cost packaging solutions, and needs to be considered at each moment of the emulation to get accurate thermal measurements for this type of MPSoC packaging technology.

8. CONCLUSIONS

MPSoC architectures have been proposed as a promising solution to tackle the complexity of forthcoming embedded systems. These future consumer devices will contain an extremely large amount of transistors, thanks to nanoscale technologies, but will be very hard to design, as they must execute multiple complex real time applications (e.g., video processing or 3D games) while meeting several additional design constraints (e.g., energy consumption or short time-to-market). Moreover, the rise of temperature in the die for on-chip components can seriously affect the performance and reliability of final MPSoC designs. In this article we have presented a new HW-SW emulation framework that provides designers with a powerful tool to study the thermal behavior of MPSoC designs at three different architectural levels, namely, processing cores, memory subsystems and interconnection mechanisms. The experimental results have shown that our proposed framework obtains detailed reports of the thermal features of final MPSoC floorplans, with speedups of three orders of magnitude compared to cycle-accurate MPSoC simulators. Also, the further additions of processing cores and more complex memory architectures in our emulation framework suitably scale. Thus, almost no loss in emulation speed occurs (conversely to cycle-accurate simulators), which enables long simulations of complex MPSoCs, as thermal modeling requires. Furthermore, the real time interaction between HW emulation and SW thermal modeling through the Ethernet connection enables the application and testing of complex dynamic thermal management policies to the emulated MPSoC at runtime.

In addition, we have used our tool to evaluate different temperature-aware placement techniques that try to compensate for the heating effects of MPSoCs. Our study indicates that the significant overheads of power dissipated in long interconnects can clearly affect the overall thermal behavior of the final MPSoC, and that a uniform distribution of power sources in the die does may not necessarily produce a uniform temperature in the final chip. Hence, MPSoCs designed in the latest technology nodes require the use of tools to study their suitable placement in an early stage of system integration, according to the applications that will be executed in each final MPSoC. Also, we have illustrated the effectiveness of the presented thermal evaluation tool to rapidly study the effects of different packaging options for concrete MPSoC solutions. Our results indicate that the selection of final packaging solutions clearly depends on the thermal management techniques included in the target MPSoCs, and more costly packagings may suffer the same heating effects as low-cost ones; Thus, a demand for expensive packaging solutions cannot be justified without prior extensive thermal exploration. Finally, we have shown the versatility of our tool in various thermal libraries, and illustrated the need for different thermal models according to the implementation requirements of the target MPSoCs (e.g., high- or low-cost packaging).

ACKNOWLEDGMENT

The authors would like to thank Srinivasan Murali for his great help to validate the thermal library and Federico Angiolini for his support with XPIPESCOMPILER.

REFERENCES

AMD. 2004. Thermal performance comparison for am486dx2 and dx4 in pdh-208 vs pde-208 package. `http://www.amd.com`.

APTIX. 2003. System explore. `http://www.aptix.com`.

ARM. 2003. Arm AMBA 2 AHB Specification. `http://www.arm.com/products/solutions/AMBA_Spec.html`.

ARM. 2004a. Arm integrator application. `http://www.arm.com`.

ARM. 2004b. ARM7TDMI-STR71xF TQFP144 and TQFP64 10x10 packages—Product datasheets. `http://www.arm.com/products/CPUs/ARM7TDMI.html`.

ARM. 2002. PrimeXSys platform architecture and methodologies, white paper. `http://www.arm.com/pdfs/ARM11%20Core%20&%20Platform%20Whitepaper.pdf`.

BENINI, L., BERTOZZI, D., BOGLIOLO, A., MENICHELLI, F., AND OLIVIERI, M. 2005. Mparm: Exploring the multiprocessor SoC design space with SystemC. *J. VLSI Signal Process. 41*, 2, 169–182.

BRAUN, G., WIEFERINK, A., SCHLIEBUSCH, O., LEUPERS, R., MEYR, H., AND NOHL, A. 2003. Processor/Memory co-exploration on multiple abstraction levels. In *Proceedings of the DATE*.

BROOKS, D. AND MARTONOSI, M. 2001. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the HPCA*.

CADENCE. 2005. Cadence palladium II. `http://www.cadence.com`.

CHEN, G. AND SAPATNEKAR, S. 2003. Partition-Driven standard cell thermal placement. In *Proceedings of the ISPD*.

CHU, C. AND WONG, D. 1998. A matrix synthesis approach to thermal placement. *IEEE Trans. Comput. Aided Des. 17*, 11, 1166–1174.

COWARE. 2004. Convergence and Lisatek product lines.

EMULATION AND VERIFICATION ENGINEERING. 2005. Zebu XL and ZV models. `http://www.eve-team.com`.

GOPLEN, B. AND SAPATNEKAR, S. 2005. Thermal via placement in 3D ICs. In *Proceedings of the ISPD*.

HEO, S., BARR, K., AND ASANOVIC, K. 2003. Reducing power density through activity migration. In *Proceedings of the ISLPED*.

HERON ENGINEERING. 2004. Heron MPSoC emulation. `http://www.hunteng.co.uk`.

IBM. 2006. IBM packaging solutions. `http://www-03.ibm.com/chips/asics/products/packaging.html`.

JALABERT, A., MURALI, S., BENINI, L., AND DE MICHELI, G. 2004. xpipescompiler: A tool for instantiating application specific networks-on-chip. In *Proceedings of the DATE*.

JERRAYA, A. AND WOLF, W. 2005. *Multiprocessor Systems-on-Chips*. Morgan Kaufmann, Elsevier.

LÓPEZ-BUEDO, S., GARRIDO, J., AND BOEMO, E. I. 2000. Thermal testing on reconfigurable computers. *IEEE Des. Test Comput. 17*, 1, 84–91.

MAGNUSSON, P. S., CHRISTENSSON, M., ESKILSON, J., FORSGREN, D., HALLBERG, G., HOGBERG, J., LARSSON, F., MOESTEDT, A., AND WERNER, B. 2002. Simics: A full system simulation platform. *IEEE Comput. 35*, 2, 50–58.

MENTOR GRAPHICS. 2003. Platform express and Primecell. `http://www.mentor.com/products/embedded_software/platform_baseddesign/`.

NAKAMURA, Y., HOSOKAWA, K., KURODA, I., YOSHIKAWA, K., AND YOSHIMURA, T. 2004. A fast HW/SW co-verification method for SoC by using a C/C++ simulator and FPGA emulator with shared register communication. In *Proceedings of the DAC*.

NAVA, M. D., BLOUET, P., TENINGE, P., COPPOLA, M., BEN-ISMAIL, T., PICCHIOTTINO, S., AND WILSON, P. 2005. An open platform for developing MPSoC. *IEEE Comput.*, 60–67.

PACI, G., MARCHAL, P., POLETTI, F., AND BENINI, L. 2006. Exploring temperature-aware design in low-power MPSoCs. In *Proceedings of the DATE*.

PAULIN, P., PILKINGTON, C., AND BENSOUDANE, E. 2002. Stepnp: A system-level exploration platform for network processors. *IEEE Des. Test 19*, 6, 17–26.

R. W. FLOYD, E. A. 1985. Adaptive algorithm for spatial gray scale. In *Proceedings of the ISDT*.

ROHOU, E. AND SMITH, M. 1999. Dynamically managing processor temperature and power. In *Proceedings of the FDO*.

SEMICONDUCTOR INDUSTRY ASSOCIATION (SIA). 2004. The international technology roadmap for semiconductors. `http://public.itrs.net/`.

SKADRON, K., STAN, M., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2002. Thermal-RC modeling for accurate and localized dynamic TM. In *Proceedings of the HPCA*.

SKADRON, K., STAN, M. R., SANKARANARAYANAN, K., HUANG, W., VELUSAMY, S., AND TARJAN, D. 2004. Temperature-Aware microarchitecture: Modeling and implementation. *Trans. Architecture Code Optimizations 1*, 1, 94–125.

SRINIVASAN, J. AND ADVE, S. V. 2003. Predictive dynamic thermal management for multimedia applications. In *Proceedings of the ICS*.

SU, H., LIU, F., DEVGAN., A., ACAR, E., AND NASSIF, S. 2003. Full chip leakage estimation considering power supply and temperature variations. In *Proceedings of the ISLPED*. 78–83.

SYNOPSYS. 2003. Realview maxsim ESL environment. `http://www.synopsys.com`.

VANDEVELDE, B., DRIESSENS, E., CHANDRASEKHAR, A., AND BEYNE, E. 2001. Characterisation of the polymer stud grid array, a lead-free CSP for high performance and high reliable packaging. In *Proceedings of the SMTA*.

Received September 2006; revised February 2007; accepted March 2007