A Fast HW/SW FPGA-Based Thermal Emulation Framework for Multi-Processor System-on-Chip

David Atienza*†, Pablo G. Del Valle*, Giacomo Paci*, Francesco Poletti‡, Luca Benini‡, Giovanni De Micheli†, and Jose M. Mendias*
*DACYA/UCM, Avda. Complutense s/n, 28040 Madrid, Spain.
† LSI/EPFL, EPFL-IC-ISIM-LSI Station 14, 1015 Lausanne, Switzerland.
‡DEIS/UNIBO, Viale Risorgimento 2, 40134 Bologna, Italy.
{datienza, mendias}@dacya.ucm.es, pgarciav@fdi.ucm.es,
{gpaci,fpoletti,lbenini}@deis.unibo.it, giovanni.demicheli@epfl.ch *

ABSTRACT

With the growing complexity in consumer embedded products and the improvements in process technology, Multi-Processor System-On-Chip (MPSoC) architectures have become widespread. These new systems are complex to design as they must execute multiple complex applications (e.g. video processing, 3D games), while meeting additional design constraints (e.g. energy consumption or time-to-market). Moreover, the rise of temperature in the die for MPSoC components can seriously affect their final performance and reliability. Therefore, mechanisms to efficiently evaluate complete HW/SW MPSoC designs in terms of energy consumption, temperature, performance and other key metrics are needed. In this paper, we present a new HW/SW FPGA-based emulation framework that allows designers to rapidly extract a number of critical statistics from processing cores, memories and interconnection systems being emulated on a FPGA. This information is then used to interact in real-time with a SW thermal model running on a host computer via an Ethernet port. The results show speed-ups of three orders of magnitude compared to cycle-accurate MPSoC simulators, which enable a very fast exploration of a large range of MPSoC design alternatives at the cycle-accurate level. Finally, our HW/SW framework allows designers to test run-time thermal management strategies with real-life inputs without any loss in the performance of the emulated system.

Categories and Subject Descriptors

B.8 [**Performance and Reliability**]: Performance Analysis and Design Aids; C.3 [**Special-purpose and Application-based Systems**]: Real-time and embedded systems
General Terms: Design, Measurement, Performance.
Keywords: FPGA, Emulation, MPSoC, Thermal Studies.

*This work is partially supported by the Spanish Government Research Grants TIC 2002/0750 and TIN2005-5619, the Swiss FNS Research Grant 20021-109450/1, and a Mobility Post-Doc Grant from UCM for David Atienza.

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

1. INTRODUCTION

New embedded systems applications (e.g. scalable video rendering or wireless protocols) demand complex single-chip designs to meet their real-time requirements while respecting other design constraints, such as low-power or short time-to-market [1]. Thus, complete redesigns of such multi-core systems from scratch are not possible any longer. MULTI-PROCESSOR SYSTEMS-ON-CHIPS (MPSOCS) have been proposed as a promising solution [1]. Nevertheless, one major challenge is the fast exploration of HW/SW implementation alternatives with accurate estimations (e.g. performance) to tune the final MPSoC architecture in an early design stage. Furthermore, MPSoCs can experience an alarming temperature rise in future technologies [2], which increases further their complex design.

In order to explore the HW/SW interaction, cycle-accurate MPSoC simulators have been developed [3, 4], lately even including SW thermal modelling based on run-time power consumption and floorplanning information [2]. However, these complex SW environments are very limited in performance (circa 10-100 Khz) due to signal management overhead. Thus, they are not suitable for MPSoC architectural exploration solutions running complex real-life applications. Moreover, higher abstraction levels simulators attain faster simulation speeds, but lose significantly the accuracy for fine-grained architectural tuning or thermal modelling.

One alternative to cycle-accurate simulators is HW emulation. Various MPSoC emulation frameworks have been proposed [5, 6, 7]. Nevertheless, they are usually expensive for embedded design (between $100K and $1M) and not flexible enough for MPSoC architecture exploration since their baseline architectures (e.g. processing cores or interconnections) are proprietary, not permitting internal changes. Furthermore, no flexible interconnection interfaces between HW emulation and the existing SW thermal libraries exist today. Thus, thermal effects can only be verified in the last phases of the design process, when the final components are available, which can result in expensive MPSoCs redesigns.

In this paper we present a new HW/SW FPGA-based emulation framework that enables realistic thermal studies as well as power, energy and performance constraints validation in real-time in an early stage of the MPSoC integration process. First, MPSoC HW components are mapped on an FPGA and statistics are extracted from three key MPSoC architectural levels (processors, memory subsystem and interconnections), while real-life applications are executed. Second, this run-time information is sent using a standard Ethernet connection to a configurable SW thermal modelling tool running on a host PC. Third, this tool evaluates in real-time the thermal behaviour of the final MPSoC design, and returns this

information to the FPGA emulating the MPSoC design. This final step enables testing run-time temperature management strategies. Our experiments show that this HW/SW framework provides detailed cycle-accurate reports with speed-ups of three orders of magnitude compared to cycle-accurate MPSoC simulators.

This paper is organized as follows. In Section 2, we overview related work on MPSoC analysis and testing. In Section 3 and Section 4, we present the HW architecture of the emulated MPSoCs and the HW statistics extraction system included. In Section 5 we describe our SW thermal tool. In Section 6, we detail the HW/SW MPSoC emulation flow. In Section 7, we illustrate the speed and flexibility of our emulation framework for MPSoC design. Finally, in Section 8, we draw our conclusions.

2. RELATED WORK

It is widely accepted that MPSoCs represent a promising solution for complex embedded systems [1]. This has spurred research on modelling and prototyping of MPSoCs using HW and SW.

From a SW perspective, different solutions have been suggested at different abstraction levels. Fast analytical C/C++ models have been proposed to prune very distinct design options [4]. Then, transaction-level simulators in SystemC [8] have enabled more accuracy in system-level simulation at slower simulation speed (100-200 KHz). Finally, companies have developed HDL-based cycle-accurate frameworks using post-synthesis libraries from HW vendors [9, 10]. However, their simulation speeds (10 to 50 KHz) are unsuitable for MPSoC exploration with real-life size inputs. In the academic context, the MPARM SystemC framework [3] is a complete system-level simulator. It includes cycle-accurate cores, complex memory hierarchies (e.g. caches, main memories) and interconnection mechanisms (e.g. AMBA or STBus). It can extract reliable energy and performance figures, but its major shortcoming is again its simulation speed (120 Khz in a Pentium IV at 2.8 Ghz).

HW emulation is a very important alternative to overcome the SW speed problems. In industry, Palladium II [5] can accomodate very complex systems (i.e. up to 256 Mgates) and provides a complete set of statistics. However, its main disadvantages are its operation frequency (circa 1.6 MHz) and cost (around $1 million). ASIC Integrator [6] is much faster for MPSoC architectural exploration, but it is limited to up to five ARM-based cores and AMBA interconnections. [7] proposes an interesting multi-FPGA emulation framework that works in the order of MHz, but mainly aims at IPs validation and is not suited for MPSoC design exploration. In the academic world, the TC4SOC [11] emulation framework includes proprietary 32-bit VLIW cores and enables testing several Network Interfaces (NIs) protocols mapped on an FPGA. Finally, [12] uses FPGA prototyping to speed up co-verification of C/C++ SW simulators, achieving 1 MHz. However, these works do not enable extraction of statistics of the three architectural levels we propose (interconnections, memory hierarchy and processing cores), and use them to perform MPSoC run-time thermal analysis.

Regarding thermal modelling, [2] presents a thermal/power model for super-scalar architectures. It predicts temperature variations in processor components and shows effects in leakage power and performance. [13] outlines variations of 13.6 degrees across the die in embedded cores. Both works clearly prove the importance of hot spots in future high-performance systems. Based on these and similar models, system-level solutions have been proposed to reduce MPSoCs temperatures [14, 15]. Therefore, flexible cycle-accurate thermal frameworks are needed to evaluate and explore the effects of run-time temperature-management policies.

3. MPSOC HW FPGA EMULATION

The proposed MPSoC framework uses FPGA emulation as the key element to model the HW components of the considered MPSoC platform at multi-megahertz speeds, and extract detailed system statistics that can be used in our SW thermal library running in a host PC. Figure 1 shows an overview of the baseline MPSoC HW architecture, it consists of three main elements:

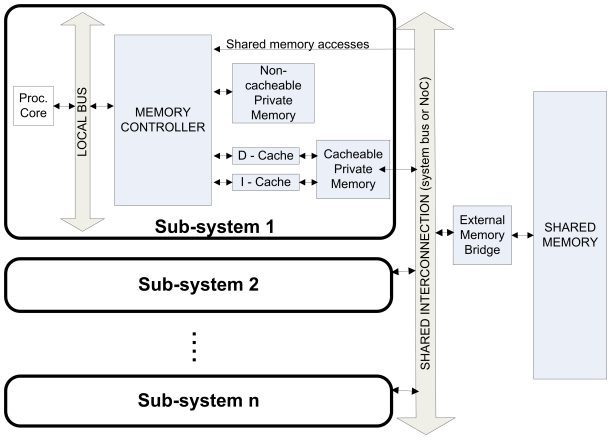


Figure 1: Overview HW architecture of emulated MPSoCs

1. Different MPSoC processing cores, such as, Power PC, Microblaze, ARM or VLIW cores.

2. The definition of configurable I-cache, D-cache and main memories (i.e. private and shared memories between processors).

3. Various interconnects, buses and Network-on-Chip (NoCs) between the first level of the memory hierarchy and main memory.

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 includes two embedded Power PCs, various types of memories (e.g. BRAM, 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.

3.1 Processing Elements

In our framework, various types of proprietary and public processing cores can be used. The accepted input forms are netlist mapping onto the underlying FPGA and HDL languages (i.e. Verilog, VHDL or Synthesizable SystemC). The addition of cores is possible since the memory controller (Subsection 3.2) includes an external pinout interface and protocol that can be modified to match the respective ones of the considered processor. Moreover, only the instruction set processing part of the core is required because its L1 memory hierarchy (e.g. caches, scratchpad) is replaced by our framework to explore different memory configurations.

In our current framework we have ported a hard-core (PowerPC 405) and a RISC-32 soft-core (Microblaze) provided by Xilinx. They only include netlist mapping and the inclusion process (pinout and protocols used) took one week. As scalability example, a complete Microblaze requires only 4% of the resources of the V2VP30 FPGA (574 out of 13.696 slices).

3.2 Memory Hierarchy

As Figure 1 indicates, in the emulated architecture two memory levels presently exist: L1 cache memories and main memories. However, additional cache levels or private memories can be added

in few minutes to each processing element, or by processor groups. This easy integration of new memory devices and protocols is possible thanks to the memory controller. One memory controller is connected to each processing core to capture all memory requests of the respective processors. Then, it forwards them to the necessary memory according to the demanded memory address. Currently, each memory controller takes 2% of the V2VP30 FPGA, and includes interfaces/protocols for four memory components and three memory address ranges:

1. Private main memory, cacheable or non-cacheable, addressable in a configurable memory range of each processor. Its size and latency are configurable, and its synthesis takes 1% of the V2VP30, apart from RAM resources that depend on the desired size.

2. Shared main memory, cacheable or non-cacheable, where its total size and latency are configurable. It uses real memories (e.g. DDR) available on the board.

3. Private HW-controlled data and instruction caches, transparent to the processors, and embedded before the cacheable address range of the two types of available main memories. It is possible to define independently for each of them their total sizes, line sizes and latencies. Currently, we support direct-mapped and set-associative caches.

Finally, each memory controller can observe different clock domains coming from multiple external interfaces. It keeps internal counters for each of the types of connected memories to keep track of the elapsed time and compare it with the user-defined latencies. The counters are reseted each time the respective memory has completed one transaction. If a memory cannot respond in the defined time, a signal is activated and sent to the VIRTUAL PLATFORM CLOCK MANAGER (VPCM) module to temporarily freeze the virtual clock of the MPSoC emulation (see Section 4).

3.3 Interconnection Mechanisms

The third configurable element in our framework is the interconnection between the memory controller and main memory. We have included both buses and NoCs, which are enabled by using a configurable main memory bridge in the device side. It includes two different public pinout interfaces: one relates to the memory and the other one to the instantiated interconnection. This enables us to extend the current list of available interconnection mechanisms by modifying the required pinout and protocol. The two Xilinx buses currently included are the On-Chip Peripheral Bus (OPB) for general-purpose devices and Processor Local Bus (PLB) for fast memories and processors. Also, we have created our own configurable (i.e. bandwidth and arbitration policies) 32-bit data/address bus for exploration purposes. Its synthesis takes 1% of the V2VP30. In addition, custom-made NoCs (e.g. number of switches and links) for our framework can be generated using XpipesCompiler [16]. We have modified the memory controller and main memory bridges to generate Open Core Protocol (OCP) transactions as Xpipes NIs require [16]. Regarding FPGA utilization, a complex NoC-based system with 6 switches of 4 input/output channels and 3 output buffers uses 70% of the V2VP30. Furthermore, as with processing cores, other proprietary bus (e.g. AMBA, STBus) can be added to the framework as blackbox, we only need to know the used protocol and external pinout.

4. STATISTICS EXTRACTION SYSTEM

The main feature pursued in the statistics extraction system is its transparent inclusion in the tested MPSoC architecture, and with minimum penalty in performance in the emulation process. Its global structure is depicted in Figure 2. We have implemented HW sniffers (Subsection 4.1) that monitor certain signals of the

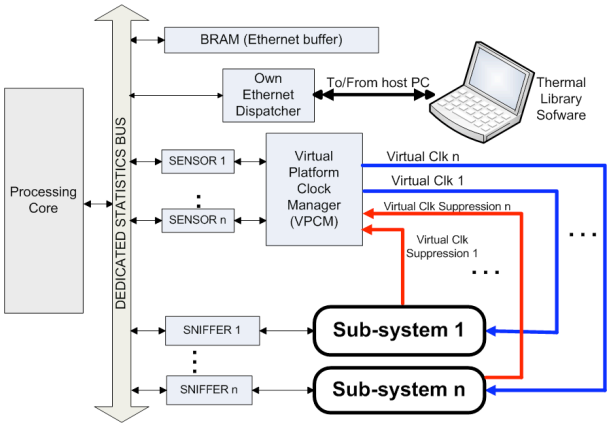


Figure 2: Overview of the statistics extraction subsystem

memory controller and external pinout of emulated MPSoC components, and stores the obtained statistics in a buffer created in FPGA BRAM memory. This buffer is concurrently processed by our Ethernet dispatcher to send MAC packets in our own format to the SW thermal modelling tool running in the connected host PC. One key additional element in this extraction mechanism is the VPCM module, which enables stopping/resuming the statistics extraction mechanism in case of congestion of the Ethernet connection with the host PC (Subsection 4.2).

4.1 HW Sniffers

The HW sniffers transparently extract the statistics from each of the MPSoC components defined in the floorplan. All sniffers have a dedicated interface to capture internal signals from the module they are monitoring and a connection to our custom statistics bus. For temperature monitoring, HW sniffers measure the time that each processor spends in active/stalled/idle mode at run-time, the number and type of accesses to each memory in the system, and the signal transitions in the buses or NoC interconnects. Also, HW sniffers are memory-mapped in the address range of the processors of the emulated subsystems. Thus, they can be de/activated at run-time through SW calls by the processors of the emulated system.

To ease the creation of new HW sniffers, there is a basic skeleton. We provide two types of sniffers. The first one, called event-logging, exhaustively logs all events that occur in the platform. The second type of sniffers, called count-logging, are designed only to count switching activity for power figures and high-level events (e.g. cache misses, bus transactions, memory accesses); Thus, generating more concise results, and what typically designers demand from cycle-accurate simulators to test their system. Our results indicate that practically an unlimited number of event-counting sniffers (i.e. floorplan cells) can be added to MPSoC designs without deteriorating the emulation speed due to signal management. This is one of the main differences with SW cycle-accurate simulation systems and enables significant speed-ups compared to them (Section 7). Finally, the overhead in the FPGA for one event-logging sniffer is 0.2% while for an event-counting sniffer is 0.3%.

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 differente types of input signals. First, the physical clock generated in the oscilator of the FPGA (not shown in Figure 2 for simplification purposes),

Table 1: Power for most important components of an MPSoC design (130nm bulk CMOS technology)

	Max. Power@100 MHz	Max. Power density
RISC 32-ARM7	5.5mW	$0.03W/mm^2$
RISC 32-ARM11	1.5W (Max)	$0.5W/mm^2$
DCache 8kB/2way	43mW	$0.012W/mm^2$
ICache 8kB/DM	11mW	$0.03W/mm^2$
Memory 32kB	15mW	$0.02W/mm^2$

which in the current implementation is set to 100 MHz. Second, one signal from each memory controller of the emulated MPSoC subsystems (VIRTUAL CLK SUPRESSION 1..N in Figure 2) 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 the different temperature sensors (SENSOR 1..N in Figure 2) that monitor if any component has increased its temperature beyond/below a certain threshold, which enables the use of run-time thermal management policies (see Section 7 for examples). The use of virtual clock domains generated by the VPCM is two-fold:

- First, the emulation of MPSoCs can be done for different physical features than those of available HW components. Once the respective (VIRTUAL CLK SUPRESSION 1..N signal is risen, the corresponding (VIRTUAL CLK signal of that sub-system (or the set of sub-systems) is activated. Hence, the stopped processor preserves its current internal state until it is resumed by the VPCM when the memory controller informs that the information requested is available in the accessed memory. This abstraction layer in the platform allows us to implement the corresponding memory resources either in internal FPGA memory (optimal performance) or with external memories (bigger size), balancing emulation performance and use of resources. For instance, if the desired latency of main memories are 10 cycles, but the available type of memory modules in the FPGA are slower (e.g. use of DDR instead of SRAMs), the VPCM will stop the clock of the processors involved at run-time, thus hiding the additional clock cycles required by the memory. Our VPCM includes two clock domains: (1) microprocessor, memories and interconnections; (2) memory controllers.

- Second, the VPCM virtual clock of all or part of the components in the emulated MPSoC can be transparently stopped/resumed at run-time in case of saturation of the Ethernet connection while downloading the extracted statistics.

The combination of these two mechanisms enables the execution and thermal modeling of HW configurations of the emulated MPSoC at a different speed than the allowed clocked speed of the 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 effects in thermal modeling 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, in case of using a 10 ms statistics sampling frequency with a desired virtual clock emulation of 500 MHz, divided by a real working frequency of 100 MHz on the FPGA, our framework will sample every 50 ms of real execution, but analyzed by the SW thermal library as representing 10 ms of actual emulated execution. The configurable SW thermal modelling is summarized next.

5. MPSOC SW THERMAL MODELLING

Our SW thermal tool is a C++ library that enables thermal exploration in silicon bulk chip systems. It can evaluate the thermal behaviour in devices modelled at different levels of abstraction (i.e. gate level, RTL level and architectural level) and its space resolution for thermal accuracy is configurable (i.e. number of temperature cells defined in a fixed area). In our case, we have set the speed to investigate the run-time thermal behavior of multiple cores and embedded memories on a single die and we look at package solutions for low-power MPSoCs, which have a much higher thermal resistance. The switching activities of the wires and the components in the die for this thermal analysis are obtained from our FPGA-based MPSoC emulation (see Section 3).

5.1 Power estimation

In Table 1, we summarize the values used for the components of our emulated MPSoC. These values have been derived from industrial power models for a 0.13 μm technology. We ignore leakage energy because in this technology the impact of leakage is very limited, particularly for low-power system design.

5.2 Thermal estimation

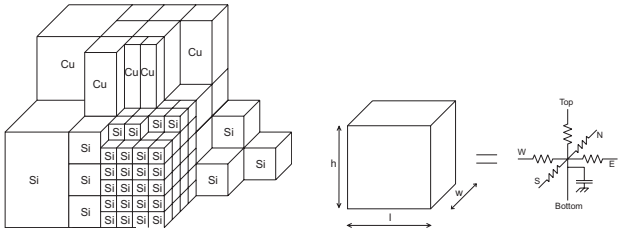
Usually MPSoCs HW are made of silicon die wrapped into a cheap package placed on a PCB. In this case the heat flow starts from the bottom surface of the die and goes up to the silicon, passes through the heat spreader and ends at the environment interface, where the heat is spread by natural convection [2]. Therefore, for modeling the heat flow, we rely on an equivalent electrical RC model similar to [2] (Figure 3(b)). However, in our case we have adopted non-linear resistances inside the silicon, in order to match the behaviour of thermal conductivity. Then, we consider the heat spreader made of copper and use linear resistances (see Table II).

Table 2: Thermal properties

silicon thermal conductivity	$150 \cdot \left(\frac{300}{T}\right)^{4/3} W/mK$
silicon specific heat	$1.628e-12 J/um^3 K$
silicon thickness	$350um$
copper thermal conductivity	$400 W/mK$
copper specific heat	$3.55e-12 J/um^3 K$
copper thickness	$1000um$
package-to-air conductivity	$20K/W$ in low power

The die and heat spreader are divided in cubic shape cells of several sizes (see Figure 3(a)). This way we can place the smallest cells in the crucial points of the studied MPSoC to obtain high resolution and insert larger ones where the conditions are not critical. Each cell has five thermal resistances and one thermal capacitance, four resistances model horizontal thermal spreading and the fifth one covers the vertical thermal behaviour. The generated heat is modelled by adding an equivalent current source to the cells on the bottom surface. The heat injected by the current source corresponds to the power density of the architectural component covering the cell (e.g. memory, processor) multiplied by the surface area of the cell. No heat is transferred down into the package from the bottom cells. Then, the heat from the top surface cells is removed through natural convection, which is modelled by connecting an extra resistance in series with their resistance. This resistance value is equal to the package-to-air resistance weighted with the area of the cell to the area of the spreader. We use 20K/W package-to-air resistance, higher than those published by package vendors, because of the uncertainty of final MPSoC working conditions. Finally, each cell interacts only with its neighbours, which results in a linear complexity problem with respect to the number of used cells. Currently, we can analyse 2 seconds of simulation (in a 660-cell floorplan), in 1.65 seconds on a Pentium 4 at 3Ghz, which is fast enough to interact in real-time with our FPGA-based MPSoC emulation.

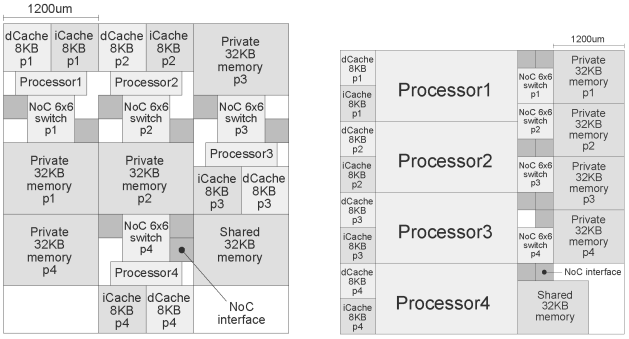
The thermal model was calibrated against a 3D-finite element



(a) (b)

Figure 3: (a) Chip divided in cells; (b) Equivalent RC circuit

analysis given by an industrial partner. In our experiments we have used two floorplans: (a) 4 ARM7 core at 100 MHz; (b) 4 ARM11 at 500 Mhz, both in 130nm technology (Figure 4).



(a) (b)

Figure 4: MPSoC floorplan with (a) 4 arm7 cores and (b) 4 arm11 cores

6. HW/SW MPSOC EMULATION FLOW

Our fast system exploration flow of MPSoC designs with thermal management, depicted in Figure 5, combines the statistics extraction from HW emulation and SW thermal simulation of MPSoCs. First, the HW and SW MPSoC components are defined. The user specifies one concrete HW architecture and the respective HW sniffers to extract statistics for three architectural levels (processing cores, memory subsystem and on-chip interconnections) using predefined HDL modules and sniffers from our repository (Section 3). No effort is required for the designer to use the memory hierarchy and interconnection mechanisms (e.g. generation of OCP transactions) since they are generated by the underlying emulated HW architecture. Also, in this phase it is compiled the application/s that will be executed in the emulated MPSoC. Currently, we use the GNU C (`gcc`) and C++ (`g++`) compilers/linkers from Xilinx EDK tool for Power PC and Microblaze cores. EDK can load different binaries on each processor. This complete HW synthesis/SW compilation phase requires 10-12 hours for a complex MPSoC architecture with 8 processors and 20 additional HW modules. Moreover, resynthesis after modifications in core configurations takes less than 1 hour, and compiling extra applications only few minutes.

In the next phase the floorplanning to be evaluated is defined according to the emulated MPSoC. At this moment the different energy and frequency values for each HW MPSoC component is set for the technology to be explored. Also, the granularity of temperature updates and communication between the FPGA and the SW thermal model is fixed (in our experiments it was $10ms$, Section 7).

Later, the whole HW emulated MPSoC is uploaded onto the Xilinx FPGA using a JTAG device and our SW thermal model is

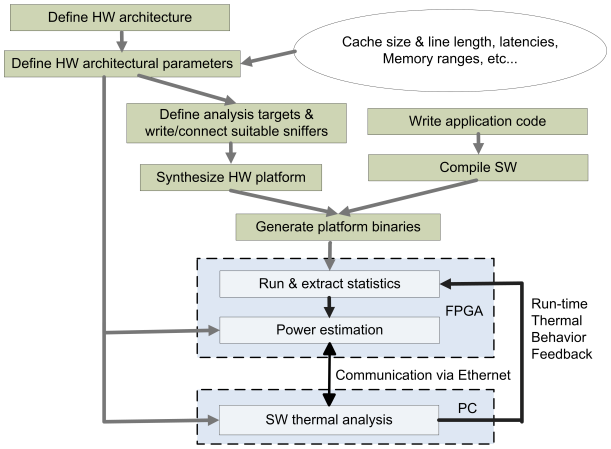


Figure 5: Complete HW/SW flows included in the FPGA-Based emulation framework

launched in a host PC, and both communicate via an standard Ethernet connection. Then, our whole framework runs autonomously. While the emulated system is running, the statistics about the power values for each layout cell are concurrently extracted, and sent to the thermal simulator running onto the host PC. This simulator calculates in real-time the new temperatures for each cell and feeds them back using MAC packets to the FPGA-based emulation. Finally, according to this new received information, the implemented temperature manager in our FPGA updates can apply concrete runtime thermal management policies (see Section 7 for an example).

7. EXPERIMENTAL RESULTS

We have compared the performance and flexibility of the HW/SW emulation framework with the MPARM framework [3] and its SW thermal library by running intensive MPSoCs processing kernels. MPARM is executed on a Pentium IV at 3.0 Ghz with 1 GB SDRAM.

First we have evaluated the speed-ups of the HW/SW emulation framework for MPSoC architecture exploration, without thermal modelling, in comparison to cycle-accurate SW simulators. We tested various configurations of interconnections and processors (1 to 8) using a complex L1 hierarchy for each core with 4 KB D-cache/I-cache, 16 KB of private memory, and a global 1-MB main shared memory. All processors use OPB and OCP buses. As 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 have explored the use of NoCs [16] instead of buses. The tested NoC had 2 32-bit switches with 4 inputs/outputs and 3-package buffers. This NoC-based MPSoC required 80% of our FPGA. As SW drivers, first, we have used a kernel application (MATRIX in Table III) that performs independent matrix multiplications at each processor private memory and combined in memory at the end. Second, we have used a dithering filtering (DITHERING in Table III) using the Floyd algorithm [17] in two 128x128 grey images, divided in 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 better than SW simulation. In fact, the exploration of MPSoC solutions with 8 cores for the Matrix driver took 0.18 seconds per run in our case, but 155 seconds in MPARM (at 125 KHz), resulting in

Table 3: Timing Comparisons between our MPSoC emulation framework and MPARM

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

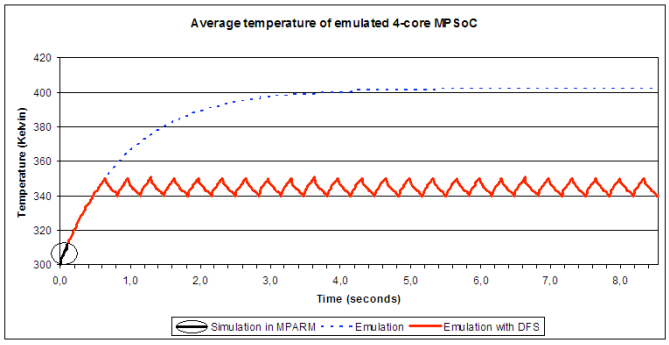


Figure 6: Temperature evolution of Matrix-TM at 500 MHz

a speed-up of 664×. Moreover, the exploration of NoCs with complex SW drivers (Dithering with 4 cores, 30 HW MPSoC components), shows larger speed-ups (860×) due to signal management overhead in cycle-accurate simulators (Table III).

Next, we have tested the real-time interaction between the FPGA emulation and the SW thermal library. We have defined a system with 4 RISC-32 processors including 8KB direct-mapped instruction/data caches and a 32KB cacheable private memory. One 32KB shared memory exists in the system and the interconnection utilized is a NoC of 4 6x6-switches 6. The floorplan includes 28 thermal cells (Figure 4) and the dimensions of NoC components were obtained after building a layout. The dimensions and energy figures of memories and processors are shown in Table 1 and were provided by an industrial partner. As SW driver, we have defined a workload of 100K matrices in the Matrix benchmark (MATRIX-TM in Table III and Figure 6) to stress the MPSoC processing power and observe thermal effects. The obtained timing results (Table III) show that our HW/SW emulation framework takes 5 minutes approximately for the whole execution of the driver, including thermal monitoring, versus 2 days in MPARM for just 0.18 sec of real execution (left corner on Figure 6); Thus, the framework achieves more than three orders of magnitude of speed-ups (1612×) compared SW-based thermal simulation, making feasible to study in a reasonable time long thermal effects.

Finally, we 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 6. They indicate the need to perform long emulations to estimate thermal effects (note in Figure 6 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 explore the possible benefits of run-time thermal management within our HW/SW emulation framework. To this end, we have implemented a simple threshold monitoring policy using the available HW temperature sensors. The policy consists in a simple dual-state machine that monitors at run-time if the temperature of each MPSoC component increases/decreases above/below two certain thresholds that we have defined (350 or 340 degrees Kelvin). Then, the temperature sensors inform the VPCM, which performs DYNAMIC FREQUENCY SCALING (DFS) choosing 500 or 100 MHz accordingly. The results are shown in Figure 6 and indicate that this simple thermal management policy is highly beneficial in this concrete MPSoC design. Furthermore, it outlines the potential benefits of HW/SW emulation to explore the design space of complex thermal management policies in MPSoCs, compared to SW cycle-accurate simulators that suffer from important speed limits.

8. CONCLUSIONS

MPSoC architectures have been proposed as a possible solution for forthcoming embedded systems. In this paper we have presented a new HW/SW FPGA-based emulation framework that enables the rapid extraction of a large range of statistics, including thermal modelling, at three different architectural levels of MPSoC designs, i.e. processors, memory subsystem and interconnection mechanisms. The experimental results have shown that our proposed framework including thermal modelling obtains detailed reports with an speed-up of three orders of magnitude compared to cycle-accurate MPSoC simulators. Also, almost no loss in emulation speed occurs when more processors and complex memory architectures are added, conversely to cycle-accurate simulators. This enables long simulations of MPSoCs as thermal modelling requires. Finally, real-time interaction between HW emulation and SW thermal modelling is possible using a standard Ethernet connection. Therefore, the returned temperature feedback enables testing run-time thermal management policies in emulated MPSoCs.

9. REFERENCES

[1] A. Jerraya, et al. *Multiprocessor SoCs*. Morgan Kaufmann, 2005.

[2] Kevin Skadron, et al. Temperature-aware microarchitecture: Modeling and implementation. *TACO*, pp. 94–125, 2004.

[3] L. Benini, et al. Mparm: Exploring the MPSoC design space with SystemC. *Journal of VLSI*, pp. 169-182, 2005.

[4] G. Braun, et al. Processor/memory co-exploration on multiple abstraction levels. In *Proc. of DATE*, 2003.

[5] Cadence Palladium II, 2005. `http://www.cadence.com`.

[6] ARM integrator AP, 2004. `http://www.arm.com`.

[7] Emulation and Verification Engineering. Zebu Xl and ZV models, 2005. `http://www.eve-team.com`.

[8] ARM. PrimeXSys platform architecture and methodologies, white paper. Technical report, 2004.

[9] M. Graphics. Platform express and primecell, 2003. `http://www.mentor.com/`.

[10] Synopsys. Realview Maxsim ESL environment, 2003. `http://www.synopsys.com/`.

[11] M. Diaz Nava, et al. An open platform for developing MPSoCs. *IEEE Computer*, pp. 60–67, 2005.

[12] Y. Nakamura, et al. A fast HW/SW co-verification method for SoC by using a c/c++ simulator and FPGA emulator with shared register communication. *Proc. DAC*, 2004.

[13] H. Su, et al. Full chip leakage estimation considering power supply and temperature variations. *Proc. ISLPED*, 2003.

[14] J. Srinivasan, et al. Predictive Dynamic Thermal Management for Multimedia Applications. *Proc. HPCA*, 2001.

[15] M. Huang, et al. A framework for dynamic energy efficiency and temperature management. *Proc. MICRO*, 2003.

[16] A. Jalabert, et al. xpipescompiler: A tool for instantiating application specific NoCs. *Proc. DATE*, 2004.

[17] R. W. Floyd, et al. Adaptive algorithm for spatial gray scale. *Proc. ISDT*, 1985.