View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

2009 IEEE Computer Society Annual Symposium on VLSI

Inducing Thermal-Awareness in Multicore Systems Using Networks-on-Chip

Emilio Martinez⋆, David Atienza†
⋆DACYA - Complutense University of Madrid,
C/ Jose G. Santesmases, 28040 Madrid, Spain. E-mail: emmartin@pdi.ucm.es
† Embedded Systems Laboratory (ESL) - EPFL,
EPFL-STI-IEL-ESL, 1015 Lausanne, Switzerland. E-mail: david.atienza@epfl.ch *

Abstract

Technology scaling imposes an ever increasing temperature stress on digital circuit design due to transistor density, especially on highly integrated systems, such as Multi-Processor Systems-on-Chip (MPSoCs). Therefore, temperature-aware design is mandatory and should be performed at the early design stages. In this paper we present a novel hardware infrastructure to provide thermal control of MPSoC architectures, which is based on exploiting the NoC interconnects of the baseline system as an active component to communicate and coordinate between temperature sensors scattered around the chip, in order to globally monitor the actual temperature. Then, a thermal management unit and clock frequency controllers adjust the frequency and voltage of the processing elements according to the temperature requirements at run-time. We show experimental results of the infrastructure to implement effective global temperature control policies for a real-life 4-core MPSoC, emulated on an FPGA-based emulation framework.

1. Introduction

When considering embedded *MultiProcessor Systems-on-Chip* new challenges arise for temperature management. These systems are subjected to temperature unbalancing and heat flow problems [10, 12]. Thus, solutions for early stages of design flows have been recently proposed [11]. However, these static approaches cannot handle heterogeneous application scenarios targeted by embedded MPSoCs with multimedia applications [15, 1, 2].

On the other side, these systems offer new degrees of freedom to implement runtime temperature control strategies based on frequency allocation of each core or workload migration [13, 9]. However, despite the large number of thermal-aware policies proposed so far, there is lack of understanding on how these policies can be implemented and supported inside a real MPSoC. Several questions are open concerning how to place temperature sensors, distribute their information and implement decision policies.

In this work we address these issues by presenting a temperature-management hardware infrastructure that exploits the NoC to propagate temperature-related information collected by distributed thermal sensors. This information is conveyed to a centralized *Thermal Management Unit* (TMU) that dynamically configures the frequencies and voltages of *Processing Elements* (PE) depending on the application requirements using dedicated core clock managers. The built infrastructure is flexible and scalable for many-core systems, exploiting NoC's features. We used XPipesCompiler [3] as a NoC technology and we enhanced its design and synthesis tools to include the temperature management support inside the design flow. In particular, the NoC design environment exploits the existing *Network Interfaces* (NIs) to transmit the temperature messages as short control messages in the interconnection infrastructure, which results in a negligible bandwidth overhead to implement thermal control. We experimentally validate the effectiveness of the proposed infrastructure for thermal control using a Xilinx Virtex-V FPGA-based thermal emulation environment of a 4-core industrial MPSoC and implementing several *Dynamic Voltage and Frequency Scaling* (DVFS) policies, based on run-time monitoring provided by the NoC about the intercommunication activities of the processing cores of the MPSoC, and off-line characterization of the application behavior a parallel version of real-life multimedia video benchmark. Our results show a much better thermal balancing behavior of the 4-core MPSoC (reductions of almost 10 degrees on average temperature) using the proposed global NoC-based thermal control infrastructure in comparisons to local DVFS thermal control techniques applied on the processing cores. Moreover, we enhance the performance of the MPSoC by approximately 40%, while respecting the temperature constraints provided by the low-cost packaging of the original MPSoC, as well as achieving almost 45% energy savings with respect to local DVFS.

The rest of the paper is organized as follows. In Sec-

*This work is partially supported by the FPU fellowship number AP2007-00843 and the Spanish Research Grants TIN2008-00508.

978-0-7695-3684-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ISVLSI.2009.25

IEEE computer society

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on August 19, 2009 at 11:43 from IEEE Xplore. Restrictions apply.

tion 2 we summarize related work in the area of thermal management and NoC design. In Section 3 we describe the NoC-based thermal control infrastructure proposed in this work and its design flow. Then, in Section 4 we present different thermal control policies that exploit the proposed NoC-based thermal monitoring infrastructure. Next, in Section 5 we describe the experimental framework and results that validate our approach by emulating the thermal behavior of a real-life 4-core MPSoC. Finally, in Section 6 we summarize our conclusions.

2 Related work

Temperature control has been an important issue since the last two decades especially for high-end systems. Given the raise of importance of early thermal analysis, various thermal model for systems-on-chip have been developed [16, 12]. Also finite element [8] and Green-function [17] based algorithms have been applied for on-chip thermal analysis. All these models can be used at design time to study better placements of components and sizing of cooling elements. In our case, we use the model presented in [5] to accurately estimate the run-time temperature using FPGA MPSoC emulation, which enables a fast thermal emulation for long transient intervals, as MPSoC require.

Based on the previous thermal modeling tools, *Dynamic Thermal Management (DTM)* techniques have been suggested for processors using architectural adaptation, DVFS, task and activity migration and profiling-based techniques. In [12], it is proposed to use formal feedback control theory as a way to implement adaptive techniques in the processor architecture. Also, [4] performs extensive studies on empirical DTM techniques when the power consumption of a processor crosses a predetermined threshold (i.e. 24W). In [13] it is presented an approach that uses profiling to predict the theoretical highest performance within a thermally-safe DVFS configuration for multimedia workloads. We use this technique as part of the basic DTM approach we propose for NoC-based thermal control (Section 4).

Other solutions have been proposed with the scope of reducing temperature difference within the chip through activity migration [9], that is aimed at reducing peak temperature by moving computation between replicated units. Thermal-aware design for NoC-based architectures have also being recently proposed. In [10] an algorithm for the optimal placement of on-chip components is proposed for achieving thermal balancing while minimizing communication cost. This work is complementary to our approach, and can be applied in early phases of the design when the floorplan of the final chip can be modified, while our approach deals with run-time thermal control.

Finally, NoCs are envisioned as the future on-chip interconnect since they can overcome the scalability bottleneck present in traditional bus interconnects in the MPSoC domain, and several NoC topology proposals and interfaces have been proposed at different levels of abstraction [3] according to different application requirements. Recent works [11, 10] outline the benefits of using NoCs to control the power consumption or temperatures in SoCs. They rely on local information and exploit power-related messages transmitted using the on-chip network. In our approach we extend these ideas by integrating the global thermal management aspect as basic part of NoC design. Thus, the NoC acts as an active element that dynamically monitors the communication of the processing blocks and provides proactive thermal control of the MPSoC.

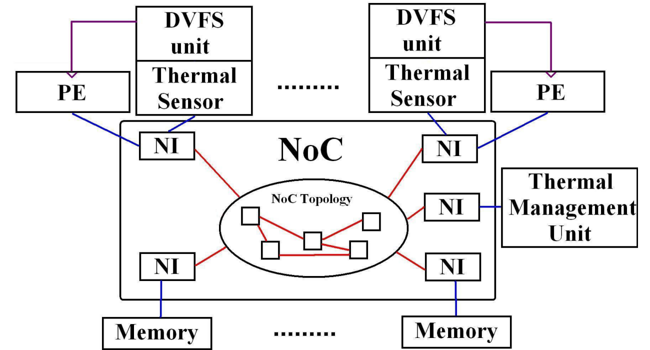

Figure 1. Active NoC-based MPSoC

3 NoC-based MPSoC Thermal Control

As depicted in Figure 1, our proposed MPSoC including the NoC-based thermal control infrastructure consists of six different components: (i) Processing Elements (PEs), (ii) On-chip memories, (iii) a regular or custom NoC, (iv) Thermal Sensors or TSs associated to PEs, (v) a global Thermal Management Unit (TMU), and (vi) a set of DVFS units.

The PEs and the memories are the basic components in regular NoC-based MPSoCs already designed for a given application. The TSs can provide the temperature of a PE to the TMU through control packets injected in the regular NIs associated to each PE. Then, the TMU is an element that first collects all the temperatures provided by the TSs of the system, and then communicates with the DVFS unit/s in order to set the frequency of the PEs of the system according to a certain global DVFS strategy. Since the MPSoC is networked, the TMU can be localized at any unit of the underlying NoC, such as an NI or a switch. Then, the TMU executes a thermal policy based on manual optimization, which has been defined at compile-time according to the possible workloads of the working environment where the NoC-based SoC will be used (see Section 4 for more details). Knowing what the distribution of the temperatures on the die is, and knowing what the workload of each processor is, it can setup the voltage and the frequency of each component in order to guarantee that a user-defined temperature threshold is not exceed and improving thermal

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on August 19, 2009 at 11:43 from IEEE Xplore. Restrictions apply.

balancing (see Section 4 for more details).

In this design, we have associated our TSs with PEs because our first goal is to control the temperature of PEs, but associating TSs to other components of the SoC would be also feasible. We also do not restrict ourselves to single TS per component. In fact, our proposed infrastructure can include several TSs per NI, in case a finer granularity on the thermal distribution is required. Finally, We attach our TSs around the cores because it enables keeping the complexity of the NoC design flow (see Section 3.1), as we do not need to add any channels to the NoC or increase the number of inputs/outputs of regular switches. Indeed the scalability of NoCs applies to our NoC-based thermal control infrastructure, since we do not add any extra design complexity or restrictions to the design of the NoC itself.

3.1 NoC Design Flow for Thermal Control

In this section we overview the design flow for NoC-based MPSoCs with thermal management support. Our methodology relies on SunFloor [14] and XPipesCompiler [3] to automatically generate the basic NoC topology without thermal control. Nevertheless, our approach is general and can be combined with any other state-of-the-art NoC design flow.

During the first phase of our design flow, once the NoC interconnect has been defined for the final MPSoC platform, we include our thermal management components. The first element to add is the TMU. This component can be located in any position of the NoC topology as long as it can communicate to every node (memories, PEs) of the MPSoC, where thermal control has to be applied. In our designs, the TMU unit is placed using a dedicated switch and NI, where the conflicts with incoming outgoing traffic to any other MPSoC nodes are minimized. Also, the TMU includes a dual master/slave NI to be able to access the TS as a processing element, and to receive the replies in the slave interface, and non-acknowledged write responses are used to make this communication of thermal information as fast as possible. Then, the TMU uses its master interface to send requests to set the frequencies of PEs to the DVFS units, which use a slave interface. Thus, using this master/slave scheme less than 3% of bandwidth is utilized for transmitting thermal control packets (see Section 5).

Then, the DVFS unit must be inserted in the MPSoC. Our design can support several DVFS units; thus, each DVFS unit can generate the value for one or several PEs. On the one hand, when having only one DVFS unit which drives all the PEs of the MPSoC (as in [2]), the most suitable location to place it is on the same NoC switch as the TMU. Hence, preference (i.e., the shortest possible latency) is given to the intercommunication channel between the TMU and the DVFS unit. On the other hand,when there is one DVFS unit per PE (e.g., [1, 7]), each DVFS unit is connected to the same switch used in the original NoC topology design for its respective PE. Therefore, the length of the clock signals (from DVFS units to PEs) across the die are minimized. Any other intermediate configuration regarding the number of DVFS units with respect to the number of PEs is feasible as well. In this case, each DVFS unit should be placed in the switch that minimizes the distance to all the PEs it controls. Thus, the length of clock signals on the final die are as low as possible, which reduces in turn the chances of clock skews in the final MPSoC. In this case, the inclusion of DVFS units in the SunFloor [14] design tool has been done as slave devices, since they do not require to initiate any transaction, but just receiving the requests coming from the TMU.

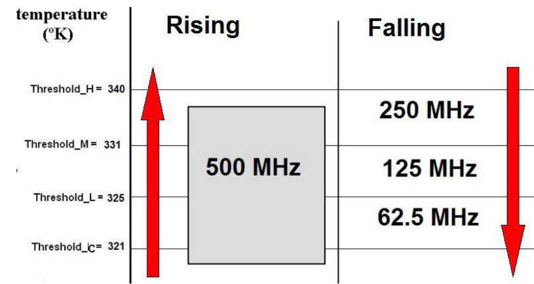

Figure 2. Three-threshold-based DVFS working points

In the final phase of the thermal control integration process, the basic NIs are reused for thermal monitoring by including a new input port to read the current temperature of the PE coming from the TS attached to it, which is sent to the TMU unit when this unit demands it, typically in the order of milliseconds (see Section 5).

4 NoC-Based Thermal Management Policies

We have implemented support in the TMU for several thermal management policies to validate the proposed NoC-based thermal control infrastructure. In general, many MPSoCs are designed to run a limited set of applications, depending on the targeted application domain. As an example, in [14], the authors show that the NoCs for industrial MPSoC designs typically support several use-cases or applications. For such MPSoCs, the workload of the system is well characterized for each of the application. Thus, using the specific processing and idle times of multimedia benchmarks (see Section 5.2 for more details), we have computed the maximum operating frequencies that needs to be supported by the processors for an application, so that the workload constraints are satisfied. However, this precomputed values do not imply that the systems is thermally optimal. Thus, we include the execution precharacterization of the applications in our TMU, such that, it can provide the support for application-level DVFS to achieve an efficient design that satisfies the thermal constraints at run-time.

To validate the functionality of our thermal control sys-

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on August 19, 2009 at 11:43 from IEEE Xplore. Restrictions apply.

tem, we have implemented three policies into the TMU for a 4-core MPSoC case study, according to different types of active NoC monitoring support of the basic MPSoC. The behavior of these policies is incremental, namely, the second includes the behavior of the first and so on. Thus, we can observe how system thermal control is enhanced when adding more local and global knowledge about the current temperature and monitoring of the transactions of the PEs in the policy implemented. In all these cases the multiprocessor system can operate at four different frequencies and voltages for each processor, namely, 500MHz, 250MHz, 125MHz and 62.5MHz, according to the figures found in the ARM 9 processing cores [7] used in our modeled 4-core MPSoC. Using simply these four operation points from the hardware management viewpoint, different DVFS-based thermal control policies can be designed to make the system more stable from a thermal viewpoint, enabling less abrupt thermal oscillations (i.e., less thermal gradients and hot-spots), by partially characterizing off-line the specific performance constraints of the set of multimedia benchmarks to be executed, similar to [13].

3-Threshold DVFS based on local temperature *(DVFS Local)*: This is an instance of the most frequently implemented policy nowadays for hardware-based thermal management in MPSoCs, where the temperature information from each PE, monitored by its respective TS, is used to provide input to a local thermal control algorithm. To keep the MPSoC temperature below a thermal run-away threshold, once the temperature of the system passes a certain temperature threshold, this policy switches gradually between different lower DVFS operation points (to prevent large DVFS changes) to reduce the temperature of a PE. Once the temperature has come down to a safe threshold, the PE start increasing their frequency of operation. In our 4-core MPSoC case study, we use three thresholds to obtain four different thermal control regions to decrease the temperature after reaching the critical rising temperature threshold, as depicted in Figure 2.

3-Threshold DVFS based on local temperature and local transactions monitoring *(DVFS and local communication)*: Besides the local temperature of the processor, as in the previous policy, this thermal management policy decides the DVFS operation point by also taking into consideration the number of outstanding transactions with the local memory of the processor. In fact, this additional information is used to understand the switch between the waiting and execution phases of the PEs in the parallel implementation of the multimedia embedded systems under execution, if a processor is not performing transactions in a certain time-out period (i.e., 10 msecs in our experiments), we assume that the processor will not make any transaction in a significant lapse of time, and its frequency is reduced to the minimum operating point (62.5MHz). On the other hand, as

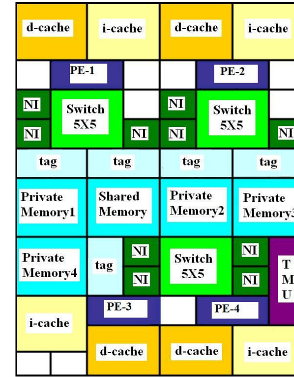

Figure 3. Floorplan of 4-core MPSoC case study using the proposed active NoC with global TMU

soon as the active NoC observes any new transaction started from that PE, the TMU starts increasing the frequency as in the previous policy. This adaptation of the frequency to the new application phase is very fast, and only requires few clock cycles, and no real performance penalty is observed in the final MPSoC.

3-Threshold DVFS with global temperature analysis and workload predictor *(DVFS and global workload predictor)*: This thermal control policy enhances the previous one by including additional global information of the multimedia application under study, which is monitored at the NoC level, as an attempt to evaluate what would be the maximum potential benefit for thermal control that can be achieved by using global run-time system behavior monitoring, achieved by the proposed active NoC infrastructure. Due to the partial regularity of the video processing of the considered multimedia application (see Section 5.2), PE-1 determines the slowest execution bound, and from a thermal balancing viewpoint, it is better to run the cores at the lowest frequency possible to finish its processing when it needs to be regrouped for the final output by PE-1, instead of finishing the computation of a certain PE at a higher DVFS operation point and then remain idle. Second this policy adapts the DVFS operation point for each PE according to a predicted time estimating when PE-1 will start collecting the data of each of the other cores in each iteration of the application under study.

5 Experimental Setup

In order to validate our NoC-based thermal control infrastructure, we have emulated a real-life 4-core ARM9 MPSoC design (see Figure 3) with low-cost plastic packaging (typical of embedded multimedia MPSoC platforms) using an FPGA-based thermal emulation platform, inspired from the structure presented in [5].

The multi-task MPSoC application used to test our NoC-based thermal control infrastructure was the *Visual Tex-*

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on August 19, 2009 at 11:43 from IEEE Xplore. Restrictions apply.

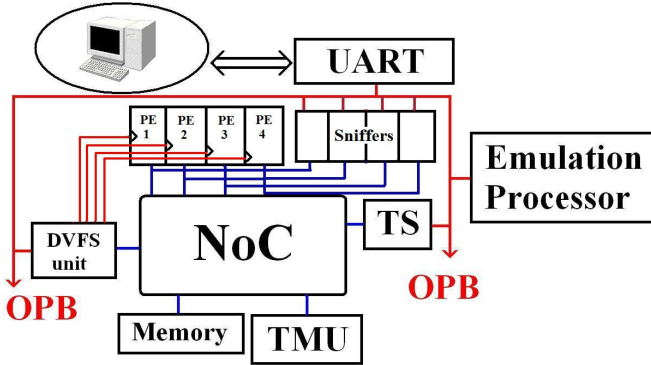

Figure 4. MPSoC Thermal Emulation Setup

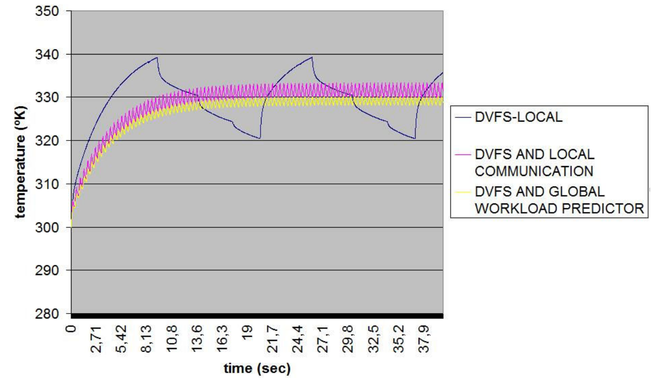

Figure 5. System temperature for each policy

ture Coding (VTC) software, which is used in the MPEG-4 standard [6] to compress the texture information in photo-realistic 3D models. As the texture in a 3D model is similar to a still picture, the application can also be used for compression of still images. It is based on the discrete wavelet transform, scalar quantization, zero-tree coding and arithmetic coding. Its software realization requires around 10K lines of C++ code. In particular, we used a parallel implementation of this benchmark based on multiple complex 32x32 bi-dimensional multiplications divided into 4 processors, 8 rows per processor, between matching windows of two consecutive frames.

5.1 Thermal Emulation Flow

We show in Figure 4 an overview of the instantiation of the thermal emulation environment implemented onto a Xilinx Virtex-V FPGA for the 4-core MPSoC. The instantiation of our emulation environment is created in three steps. First, we define the hardware part of the MPSoC emulator. It implies synthesizing the NoC topology, assembling the basic components (PEs, memories, etc.) of the emulated MPSoC and adding the TMU and the DVFS unit. We plugged our TSs on the same NI of the PEs through a shared port, as explained in Section 3.1. Our TSs are implemented as registers, which can be written by the emulation control processor and can be read by the TMU through NoC transaction requests as a master.

Second, we compile and download the software application that has to be executed by the PEs (Section 5.2). Then, to update the temperature in the TS at run-time, we used the model described in [5], which relates the temperature of the PEs and the memories by monitoring the traffic generated or received by each element of the MPSoC. To this end, we installed hardware sniffers for the PEs. These sniffers report energy consumption values of basic MPSoC components that are then analyzed by a host workstation (see Figure 4) connected to the FPGA emulation framework through a serial port. The external workstation computes then the temperature of the different parts of the MPSoC at run-time, according to the model given in [5], and the emulation control processor takes care of updating the temperature in the TSs at run-time. Then, when the update of the temperature in the TSs is performed, the emulation control processor freezes the emulation by applying a global clock gating to the complete MPSoC. This FPGA-based MPSoC emulation framework allows us to instantiate and perform long thermal behavior explorations of MPSoCs very fast with respect to HDL simulators, as shown in [5]

Finally, in the third step we define in the TMU the thermal management policy to evaluated. Instead of developing a hard-coded TMU, we have used a standard soft-core processor provided by the Xilinx Virtex-V FPGA (i.e., a MicroBlaze) to implement the policies presented in Section 4 in a faster way. Since these policies are emulated as software in our MPSoC emulation environment, it is highly configurable and can easily evaluate variations of very complex thermal control algorithms.

5.2 Experimental Results

From our experiments we first report the run-time thermal evolution of the MPSoC case study while executing multiple instances of the parallel VTC software application. As shown in Figure 5, the two thermal control schemes with active NoC monitoring using the global TMU achieve a much smoother transition in temperature variations on the die with respect to hardware-based local DVFS policies, which in turn produces a much better thermal balancing effect in the MPSoC, i.e., the temperature is stabilized in regular working conditions to approximately 328-degree Kelvin. Hence, the active NoC achieves an average working temperature reduction in the 4-core MPSoC of 10 degrees on average in comparison to local DVFS, which subsequently improves system reliability, as this factor decreases exponentially with working temperature [12].

In addition, thanks to the more balanced thermal behavior achieved by the active NoC with global TMU, the MPSoC does not make frequent transitions between extreme DVFS working points (500MHz and 62.5MHz), conversely to the local DVFS thermal control policy. Therefore, as

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on August 19, 2009 at 11:43 from IEEE Xplore. Restrictions apply.

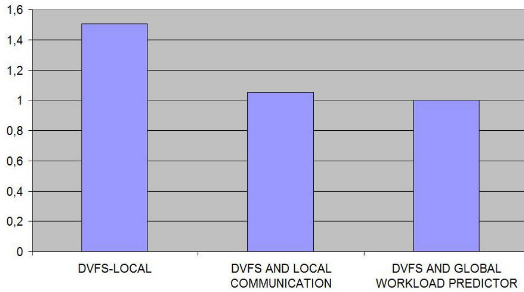

Figure 6. Energy consumption of 4-core MP-SoC using different thermal control policies (normalized to active NoC-based with global workload predictor)

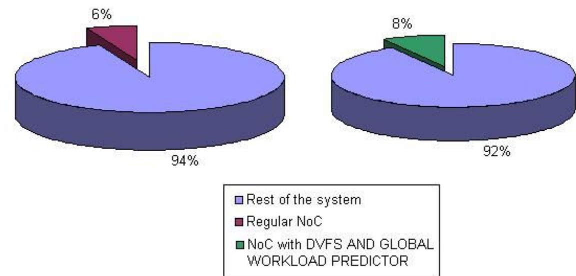

Figure 7. Comparisons of energy consumption of basic NoC and active one in 4-core MPSoC

shown Figure 6, the 4-core MPSoC experiences considerable energy consumption savings (almost 50%) using the NoC-based thermal control infrastructure with respect to the local DVFS thermal control. Also, the MPSoC achives a higher overall performance level (i.e., 40%).

Finally, we report in Figure 7 the percentage of energy consumed by the active NoC infrastructure with respect to the basic NoC interconnect using three 5x5 switches for the 4-core MPSoC (Figure 3). As Figure 7 shows, the active NoC with a global TMU only generates a negligible 2% energy overhead with respect to the basic NoC. This is due to the fact that the thermal control messages generated by the active NoC require a very limited NoC bandwidth, because the thermal dynamics of the material are quite slow (in the order of milliseconds) with respect to the volume and frequency of regular interconnect messages of the basic MPSoC components. Hence, the thermal information used by the TMU to make its thermal control decisions, as well as the thermal control messages sent by the TMUs to the PEs, can be included in the existing NoC bandwidth of the basic NoC design, as described in Section 3.1. These results illustrate the applicability of our active NoC-based infrastructure for thermal control of MPSoC architectures.

6 Conclusions

Due to the improvements in process technology, MPSoC architectures are a promising solution to provide the required performance of latest multimedia applications. However, process technology scaling imposes important temperature stress in circuit design due to increasing power density. In this paper we have presented a thermal management hardware infrastructure that exploits the concept of an active NoC interconnect to monitor the communication activity and temperature evolution of basic MPSoC components (processors, memories, etc.) and includes a centralized TMU that provides global DVFS-based thermal control policies. This proposed design flow and active NoC implementation reuses the existing NIs to transmit the temperature messages, as short control messages, in the interconnection infrastructure, which results in a negligible bandwidth overhead to implement thermal control. Our experimental results emulating a 4-core industrial MPSoC have shown that this approach achieves more than 10 degrees reduction on average working temperature than multi-threshold-based local DVFS thermal control. Moreover, as a consequence of the better thermal control, the MPSoC performance improves by almost 40%, while respecting the temperature constraints, and more than 45% energy savings are achieved.

References

[1] Cradle technologies: Multi-core DSPs for IP network surveillance, 2005. `www.cradle.com/`.

[2] ARM 11 - mpcore, 2004. `http://www.arm.com/products/CPUs`.

[3] G. De Micheli, et al. Networks on Chips: Technology and Tools. *Morgan Kauffman*, 2008.

[4] David Brooks, et al. Dynamic thermal management for high-performance microprocessors. In *Proc. of HPCA*, 2001.

[5] David Atienza, et al. HW-SW emulation framework for temperature-aware design in MPSoCs. *ACM TODAES*, August 2007.

[6] Irak Sodagar et al. Scalable wavelet coding for synthetic and natural hybrid images. *IEEE Trans. CSVT*, March 1999.

[7] i.MX31-serie Multimedia Processors, 2003. `www.freescale.com/imx31`.

[8] B. Goplen, et al. Efficient thermal placement of standard cells in 3d ics using a force directed approach. In *Proc. ICCAD*, 2003.

[9] S. Heo, et al. Reducing power density through activity migration. In *Proc. ISLPED*, 2003.

[10] W. Hung, et al. Thermal-aware IP virtualization and placement for NoC architecture. In *Proc. ICCD)*, 2004.

[11] W. Hung, et al. Thermal-aware allocation and schedulilng for systems-on-chip. In *Proc. DATE*, 2005.

[12] K. Skadron, et al. Temperature-aware microarchitecture: Modeling and implementation. *IEEE TACO*, January 2004.

[13] J. Srinivasan, et al. Predictive dynamic thermal management for multimedia applications. In *Proc. ICS03*, 2003.

[14] S. Murali, et al. Designing Application-Specific Networks on Chips with Floorplan Information In *Proc. ICCAD*, 2006

[15] ST Nomadik multimedia processor, 2004. http://www.st.com/stonline/prodpres/dedicate/proc/proc.htm.

[16] T.-Y. Wang, et al. 3-d thermal-adi: A linear-time chip level transient thermal simulator. *IEEE T-CAD)*, December 2002.

[17] Y. Zhan, et al. Fast computation of the temperature distribution in VLSI chips using the discrete cosine transform and table look-up. In *Proc. ASPDAC*, 2005.

Authorized licensed use limited to: EPFL LAUSANNE. Downloaded on August 19, 2009 at 11:43 from IEEE Xplore. Restrictions apply.