2010 IEEE Annual Symposium on VLSI

Adaptive Task Migration Policies for Thermal control in MPSoCs

David Cuesta[1], José L. Ayala[1], José I. Hidalgo[1], David Atienza[2], Andrea Acquaviva[3] and Enrico Macii[3]
[1]Complutense University, Madrid, Spain
{dcuestag@pdi, jayala@fdi, hidalgo@fis}.ucm.es
[2]Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
david.atienza@epfl.ch
[3]Politecnico di Torino, Turin, Italy
{andrea.acquaviva, enrico.macii}@polito.it

***Abstract*—In deep submicron circuits, high temperatures have created critical issues in reliability, timing, performance, coolings costs and leakage power. Task migration techniques have been proposed to manage efficiently the thermal distribution in multi-processor systems but at the cost of important performance penalties. While traditional techniques have focused on reducing the average temperature of the chip, they have not considered the effect that temperature gradients have in system reliability. In this work, we explore the benefits of thermal-aware task migration techniques for embedded multi-processor systems. We propose several policies that are able to reduce the average temperature of the chip and the thermal gradients with a negligible performance overhead. With our techniques, hot spots and temperature gradients are decreased up to 30% with respect to state-of-the-art thermal management approaches.**

I. Introduction

Recent works have demonstrated that large temperature variations cause low reliability and they also impact leakage current. Temperatures over a threshold in localized areas of the chip (hot spots) can produce timing delay variations, transient reduction in overall system performance or permanent damages in the devices [1].

The reliable and efficient functioning of MPSoCs can be satisfied by guaranteeing the operation below a temperature threshold and power budget. It is in this control problem where thermal management and balancing policies come into play. Task and thread migration policies can be proposed to manage the thermal profile in embedded multi-processor systems [2]. While traditional dynamic thermal management (DTM) techniques have already been applied, they have not considered the spatial and temporal gradients that determine the mean-time-to-failure of the devices.

Thermal simulation of MPSoCs, where the exploration of the interaction between the hardware architecture and the software layer that performs the task migration is crucial, can take an unaffordable time. Thus, in order to explore the HW/SW interaction, FPGA-based thermal emulators have been developed [3], [4]. The experimental work carried out in this work is also developed for an FPGA-based MPSoC emulation platform [5] that speeds up the simulation time and provides high flexibility in the thermal analysis.

Thus, this paper focuses on the design an implementation of three different task migration policies that are able to minimize the average temperature in MPSoCs as well as the spatial and temporal variations of the thermal profile. Our results show that they reduce the impact on the system performance to a minimum as compared to previous published approaches [5], [2], [6]. The specific contributions of our work are the following:

- three novel task migration policies based on adaptable weighted functions of three different factors: average thermal deviation between processors, maximum temperature of the overall chip and thermal gradient between cores.
- the proposed policies minimize the peak temperature and thermal gradients by considering a floorplan-aware task migration approach, at the same time as the time history of thermal gradients and thermal deviation of the different processors.
- the reliability of the system is improved by a combined minimization of time-based thermal unbalance (thermal cycles) and space-based thermal variations (hot spots).
- the experiments has been developed on a realistic MPSoC emulation platform [5], and the policies have been embedded in a multi-processor OS to assess its real-life task migration overheads in performance and temperature profile.

II. Related Work

Load balancing techniques have been studied for general purpose parallel computers in the last decade [7], [8]. However, embedded systems and MPSoCs impose constraints, as the low-cost packaging and the portability, that make necessary to develop new techniques.

Barcelos et al. [9] proposed a hybrid memory organization approach which supports the task migration algorithms with low-energy consumption constraints. In this approach, the data to be migrated can be provided either by the source node or from the shared memory.

In the area of temperature optimization, several approaches have been proposed. Donald et al. [10] introduced several thermal management policies such as dynamic voltage and frequency scaling (DVFS) and thread migration based on current temperature, but their work do not consider the thermal history of the cores. This information gives a meaningful information about the future behavior of the system and can be exploited to improve the results of the migration.

In [11], Yang et al. showed an execution ordering approach that swaps hot and cool threads in cores to control the temperature. This can only be applied once the application has been profiled.

Finally, in [5] it is proposed a heuristic optimization for thermal balancing in MPSoCs that adapts the current workload of the cores using DVFS and task migration, according to the standard deviation of the hottest and coldest cores at each moment in time during the execution. Although it shows clear benefits for thermal balancing with respect to previous thermal runaway approaches [10], it can still produce significant thermal unbalance in non-stable working conditions. (i.e., periods of small tasks being executed in the MPSoC or tasks being stop due to I/O processes) as we show in Section V), because it does not take into account the recent thermal history of the system but just the instant thermal unbalance.

Our work outperforms previous approaches with the provision of three task migration policies that optimize the thermal profile of MPSoCs by balancing dynamically the weight of the on-chip

This work has been partially funded by the Spanish Ministry under contract TIN2008-00508

978-0-7695-4076-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ISVLSI.2010.39

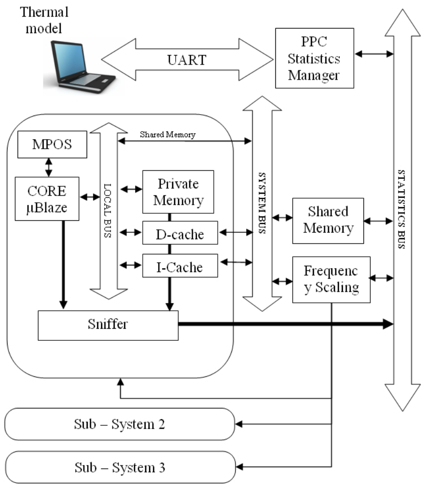

Fig. 1. Schematic view of the emulation platform.

thermal gradients, maximum temperature and effect of underlying floorplan on heat dissipation properties of each core. Moreover, the proposed policies are able to minimize the risk of system failure by the minimization of temperature-driven reliability factors, as it considers thermal unbalance in time and space, as they keep a history of the thermal profile of the target MPSoC, which minimizes the number of task migrations.

III. Emulation Platform

The thermal analysis conducted in this work requires an efficient mechanism to evaluate the performance and thermal statistics of the multi-processor system. The accuracy and the fast emulation of the system are the main constraints for the platform. Also, it is needed an MPOS that implements and manages the task migration policies.

In this work, we have used a complete FPGA-based estimation framework, implemented in a Virtex II pro VP30 and based on [4]. Figure 1 shows an schematic view of this emulation platform detailing a single core system. Using this framework we can retrieve the memory and processor statistics required by the thermal model and the migration policies (power consumption, memory misses and memory hits) by mean of hardware sniffers. This platform also includes a complete MPOS and task migration support library between the three cores of the emulated MPSoC (see Figure 5).

In this emulation platform, the collected statistical data are sent to the host PC through the serial port. In the multiprocessor system, a dedicated PowerPc is the one in charge of processing and sending the statistics to the host PC. The host translates the received information into temperature values by means of a thermal library. This thermal library splits the floorplan of the emulated system in unitary cells, which are modeled as simple $R_{thermal}C_{thermal}$ circuits. The resolution of the linear equations created by an RC grid provides the evolution in time of the temperature of the system [12].

The emulated architecture is an homogeneous multi-processor system with three 32-bit RISC cores and the PowerPC. These processor do not include a memory management unit (MMU) and the access to the cacheable private memories and to a non-cacheable shared memory is managed by the OS.

Each core runs a uClinux OS. This is based on a Linux 2.4 kernel for microprocessors without an MMU, but upgraded to support the interprocessor communication found in our target system. The OS implements the task migration policies based on *task-replication*. Thus, there is a replica of each task in every local OS, but only one processor at a time can execute it. This method requires a slightly larger private memory to hold the tasks and task intermediate states/data before migrations, but it speeds up the task migration phase because the memory allocation required by the replication of tasks is avoided. Then, the task migration takes place only at predefined checkpoints chosen by the programmer between phases of the streaming execution (e.g., between processing different frames).

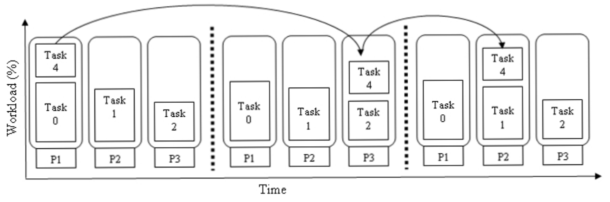

Fig. 2. Migration example between three cores.

Several modifications have been done in the OS kernel to support the floorplan-aware policy. First, the identifier and weight of the cores (used by the policies to select the candidate in the task migration, as it will be presented later) are allocated in the shared memory. Second, the OS can then access this information to apply the task migration algorithm and achieve the thermal optimization.

Finally, the emulation system has also been upgraded with a floorplan-temperature visualization tool. This tool communicates with the thermal library and, in real-time, provides a colored floorplan thermal map of the emulated MPSoC (see Figure 6). The developed tool enables a rapid inspection of the hot spots, the evolution in time of the temperature and the spatial and temporal heat spread.

IV. Adaptive And Floorplan Aware Policies For Thermal Balancing

As previously mentioned, the task migration policies that we present in this paper are devoted to reduce the thermal gradients and mean temperature in a multi-processor system, because both facts affect negatively the reliability and the leakage of the chip [1]. This assumption is even more critical for embedded systems, where the power and temperature constraints must be satisfied in parallel with requirements of high-performance execution.

The FPGA-based multi-processor platform used in our experiments has been extended with a DVFS policy as an effective way to manage the voltage and frequency settings of the cores depending on the working load. The DVFS technique implemented follows the *vertigo* policy [13]. To apply the vertigo policy a previous characterization of the tasks is needed attending to their full-speed-equivalent (FSE), defined as the load that a task imposes when it is run at full speed in a core. Therefore, if one core is running a task that loads it, e.g. 45%, the core can adapt its frequency to 45% of its maximum.

Task migration policies are proposed to balance the load in the processors and, consequently, obtain a homogeneous distribution of temperature. Figure 2 presents an example. Three cores are running four tasks with different workload. This workload in the processors is translated into temperature due to the relation with the electric activity and dynamic energy; hence, this situation will create a thermal gradient due to the unbalanced distribution of the load, being **core 1** the hottest one. Thermal balance will be achieved migrating one task from this core to one of the colder processors.

If the temperature of the chip varies slower than the rate of task migration [1], thermal balance will be achieved. In this case, we can assume that the real workload of each processor is the average of

[1]This is a common assumption because the thermal evolution is a slow diffusion process.

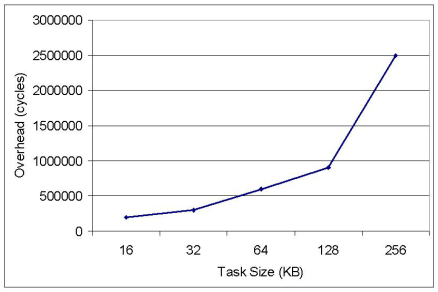

Fig. 3. Overhead of the task migration mechanism.

the total, in the example, around 55%. However, task migration must be applied carefully because it affects the performance of the system due to the overhead introduced by data transfers.

The following paragraphs analyze the state-of-the-art task migration techniques, and the policies that we propose to specifically adapt the workload of the system depending on the state of the processors.

A. Compared state-of-the-art thermal control policies

- **Enhanced Migration (Mgr)** [6]: moves the task that is running in a hot core when it exceeds a threshold temperature to the coolest core. This policy could be considered as an even improved solution of the original policy of Heat & Run, because it adds task migration at run-time, as proposed in [5], not just between stopped or starting tasks.
- **Task rotation (Rot)** [2]: inspired by a Round Robin mechanism, migrates a task between processors every time slot. This policy achieves the thermal balance in the system at the cost of an important overhead due to the frequent migrations.
- **Thermal Thresholds (Thres)**, presented in [5], moves the task running in the processor that exceeds an upper or lower threshold to a destination core. This is chosen considering the weight of the task that is going to be migrated and its impact on the workload of the processor.

B. Atomic Policies Pre-Characterization

The definition of our new task migration policies begins with the characterization of atomic policies in the multi-processor system. These atomic policies perform simple migrations only according to the temperature and the workload of the cores. The migration of the task is executed from one processor to another one with a negligible computation cost. Figure 3 shows the overhead introduced by the task replication mechanism for different sizes of the migrated task. As can be seen, the impact of migrating a 64 KB task (the one considered in our experimental work) is of 6E5 cycles, which translates into a delay of 6ms for the worst case, depending on the operating frequency (from 100 to 500 MHz) of our system. This delay could have important issues in process' deadlines for real-time tasks.

The results of the analysis of these policies are classified in several sets depending on their response to pre-defined metrics. These metrics evaluate the capability of the atomic task to reduce the thermal gradient, the maximum temperature or the mean temperature in the chip. We also performed a statistic study to classify the policies in these groups and assign a quality mark that goes from 1 (very bad response) to 5 (very good response). The granularity of the classification is enough to represent the variability expected in the results and to reflect the variations found in the metrics.

Table I shows a reduced sub-set of the atomic policies that have been considered and their classification after the statistic analysis.

TABLE I
CHARACTERIZATION OF ATOMIC POLICIES

Atomic policy	Mean Temperature	Max. Temperature	Thermal Gradient
Hot-Cold	4	5	4
Warm-Cold	2	2	1
Hot-Warm	5	4	4
Cold-Warm	1	1	1
Warm-Hot	3	3	1
Cold-Hot	1	1	2

In this table, the first column is the name of the atomic policy (it designs the origin and destination cores in the migration), being *hot* the reference for the hottest processor, *cold* for the coldest one and *warm* is the name given for those cores whose temperature is in between both hottest and coldest ones. As the goal of the analysis is the characterization of the policies, these will be always activated and the migrations will take place continuously. Finally, the initial workloads in the cores of the system are deliberately unbalanced to force the execution of the atomic policies. Next columns show the assigned quality mark for every metric.

The pre-characterization study also considered the thermal history of the cores (cores that have been cold or hot during a certain period in the past), which brought out the possibility to minimize the overhead in terms of number of migrations and amount of data transferred due to migrations.

The time window has been selected as the largest with the minimum impact on the temperature gradient after a detailed experimental study [5]. This selection of 300 ms for the time window is independent of the application run by the processors and only should be revisited in case of a new package.

C. Proposed Policies

1) Heuristic Algorithm (Heu): This algorithm is able to select efficiently among the atomic policies and achieve the thermal optimization with a minimum performance impact. The implementation of this heuristic is based on the information retrieved by the characterization phase, which provides the information about the thermal profile under the execution of the different atomic policies.

The algorithm works as follows: A time window is set and the workload and thermal information of the processors are collected at run-time during this time slot. At the end of the time window, we evaluate the data and compare them with the preferred working parameters (in terms of mean temperature, gradient and peak temperature). The atomic policy to apply is selected in order to solve the divergence of metrics between the current state and the desired one. Figure 4 shows the decision chart that explains the functioning of this heuristic.

In this figure the *Deviation* is the difference between the preferred working value (which is 50^oC for the mean temperature, 75^oC for the peak temperature and 6^oC difference for the thermal gradient) and the current state value. These values have been selected to assure a proper operation of the system. *Factor* has been tuned experimentally to balance the importance of the different decision sets, namely, giving twice more weight to the mean temperature with respect to the gradient and 1.5 more than the maximum temperature.

The proposed heuristic defines a multi-objective optimization problem. The implementation of the heuristic applies sequentially the atomic policies in case of identical unbalance in the three metrics. In this way, the complexity in the decision process is minimized to simplify the heuristic. In order to alleviate the constraint imposed by this simplified thermal controller, an adaptive policy is introduced.

2) Adaptive Policy (Adapt): This policy extends the work performed by the previous approach, collecting data at run-time and

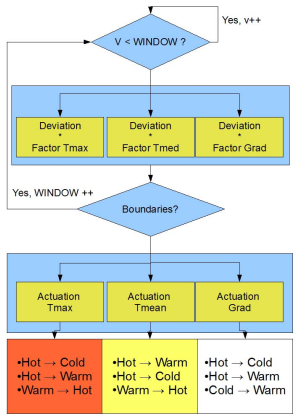

Fig. 4. Heuristic algorithm decision chart.

applying the atomic policies to achieve the optimum thermal state. This policy adapts the selection of the atomic policy by means of the statistical information of the cores, which predicts the behavior of the processors attending to the information about the past time.

This policy assigns a probability to every set of atomic policies (mean temperature, peak temperature, thermal gradient) and updates this probability every time period as follows:

$$P_t = P_{t-1} + W \tag{1}$$

$$W_{init} = M_{pref} - M_{avg} \tag{2}$$

$$W = \begin{cases} \alpha_{inc}(T_{mean}, T_{peak}, T_{gradient}) \cdot W_{init} & W_{init} > 0 \\ \alpha_{dec}(T_{mean}, T_{peak}, T_{gradient}) \cdot W_{init} & W_{init} < 0 \end{cases} \tag{3}$$

where W is the weight assigned to the sets every time period; M represents the different sets of atomic policies, as explained before; M_{pref} is the preferred working state and M_{avg} is the current state. The expressions for the increase and decrease of the probabilities are parametrized for every set of atomic policies, and the obtained probabilities are normalized in order to maintain math consistency. M_{pref} is the safe operating state already defined.

Using the previous equations, our extended OS updates the probabilities of selecting atomic policies every time window, and decides the working state by the execution of these policies. The design of the Adaptive Policy is supported by the pre-characterization of atomic policies. This initial study gives us the information of the best candidates (those atomic policies that obtain the maximum minimization of the metrics) for a task migration or task swapping in order to achieve a desired working state.

The atomic policies implemented in this adaptive technique always migrate a task from a source core to a destination core. As the temperature of the destination core is the only variable considered in the decision, more than one processor can satisfy the requirements. The last proposed policy extends the variables with the placement of the core for a more accurate selection of the destination core.

3) Floorplan-Aware Policy (FloorAdapt): This policy considers the information about the floorplan. In this way, the OS is aware of the location of the cores and accordingly selects the destination processor in a task migration. This is implemented in the kernel of the OS with the assignment of different weights to each core. The smaller this weight is, the better candidate the core is to receive tasks. This factor is calculated with the following equation:

$$G = d_{edge}^3 + \frac{1}{d_{core}^2} + d_{shared} \tag{4}$$

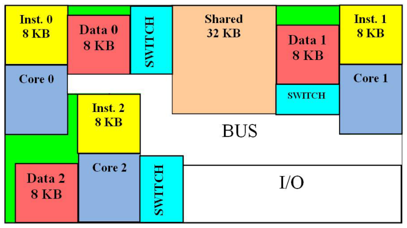

Fig. 5. Floorplan design.

where d_{edge} is the distance to the edge of the chip, d_{core} is the distance to another core (which is a heat source), and d_{shared} is the distance to the shared memory (which is a heat sink [14]). This expression has been created to resemble the strong influence of the ambient as a heat sink (cubic factor), the medium influence of the near cores as heat sources (quadratic factor) and the light influence of the shared memory as a heat sink (linear factor). The strength of the factors considers the proximity of the heat/sink and the thermal resistance of the joint.

Every time window, the thermal history of the processors is analyzed to solve possible hot spots, critical thermal gradients, or values over the safe peak temperature (75^oC). However, if the system is still working in a safe state, the task migrations will not occur and the overhead of the policies will be avoided.

The knowledge of the thermal characteristics of the cores depending on the placement is a precious information for the task migration policies. The location of the cores in the chip surface produces very different thermal behavior due to the proximity to heat sinks or heat sources which dissipate the temperature. In our floorplan design shown in Figure 5, **core 0** is close to **core 2** and both processors are prone to heat up due to the thermal diffusion from one to the other. On the other hand, **core 1** is far from the other processors but close to the edge of the chip, which increases the possibility to cool easily. Therefore, **core 1** would be selected to receive a heavy workload in case of a task migration.

The floorplan-aware policy incorporates this information about the core placement to adapt and select the probabilities of migrating or receiving a task.

V. Experimental Work

The experimental work has been conducted with the emulation platform described in section III, which has been used to model a multi-processor system with three working processors (μBlaze) and a PowerPC serving as the arbiter of the communication. The benchmark selected for the analysis is a real-life streaming application that loads the cores. The experiments have been run considering a special package derived from real-life streaming SoCs [15] for mobile embedded devices where the temperature can vary as much as 10 degrees in less than a second. The chip package has been selected to stress the number of required task migrations and, therefore, create a worst-case scenario for the validation of our techniques. Finally, the cores in the system can work at different clock frequencies under selection of the OS: 100, 200, 300, 400 and 500 MHz.

The validation of the task migration techniques has been accomplished attending to some pre-defined metrics that cover the spectrum of thermal aware optimization:

i spatial variation of the temperature of the processors: measured as the linear distance per area unit between cores at a different temperature. This metric quantifies the heat spread on the chip surface and the probability of thermal gradients.
ii mean temperature of the chip: calculated as the arithmetic mean of the processor and memory temperatures in the chip. This metric

TABLE II
INITIAL WORKING STATE.

Core (Freq.)	Load [%]	Temp. [K]
Core 0 (533 MHz)	44	340
Core 1 (533 MHz)	83	339.5
Core 2 (266 MHz)	29	328.5

relates the temperature of the devices to the energy consumption and cooling necessities.

iii maximum temperature of the chip: measured as the maximum temperature value on the chip surface. It is related with the susceptibility to temperature-driven reliability factors.

The results obtained during the validation phase have been also compared with the results provided by the policies described in Section IV.

A. Description of the Application

The software that is executed by the platform is a Software FM Defined Radio [5] application, which is a typical example in multimedia streaming. This application is composed of several tasks that can be assigned to the different processors in the system. The input data is a digitalized PCM radio signal which has to be processed in several steps to obtain an equalized base-band audio signal.

B. Evaluation of the Policies

The execution of the application in the emulation platform consists of two phases. The first one is the initialization of the OS and the tasks. As this phase does not exhibit a critical thermal state and it occurs just once during the system boot-up, the task migration policies are deactivated at this time. When this initial phase finishes, the thermal and workload state of the system is the one described in Table II. Our experimental work starts at this point setting a thermal unbalance that motivates the activation of the migration policies. In the second phase, when the execution of the application starts, all the policies described in this paper are evaluated separately.

The analysis performed for the task migration policies is two fold. Firstly, a visual inspection of the thermal distribution in the chip surface is done using the developed graphical tool. With this analysis, the evolution of temperature in real-time is obtained, as shown in Figure 6. This figure shows an example of the run-time behavior for the (a) proposed `adaptive` and the (b) `migration` [5] policies.

As shown, both policies start similarly, decreasing rapidly the presence of hot spots. However, as time evolves, the adaptive policy obtains lower temperature values and a more homogeneous thermal distribution due to the presence of short-time execution tasks. In fact, for the SDR benchmark, all the cells in the floorplan are within a range of temperature of 5 degrees when the adaptive policy is applied, while differences of more than 15 degrees can be found in certain periods for the migration policy. Similar results occur with the other task migration techniques.

Secondly, a statistical study of the distribution of temperatures in the chip under the execution of the task migration policies is accomplished. This analysis evaluates which policies have better results when applied in the multi-processor system. The mean and sigma values of the temperature for every policy are calculated in the statistic analysis and fit to a normal distribution (see Figure 7).

As can be derived from the values in the Figure, the best results in terms of thermal distribution and absolute values are achieved with the three policies specifically proposed in this paper. In particular, the adaptive algorithm concentrates the temperature of the cells within a small range of temperatures centered in the mean temperature (mean temperature 319.038 with a σ of only 2.53). The curves for the three proposed policies present: lower mean value (translated into a decrease in the average temperature of the chip) and narrower shape of the curve (translated in a smaller sigma and, therefore, a decrease in the thermal gradient of up to 30% with respect to state-of-the-art techniques [2], [5], [6]).

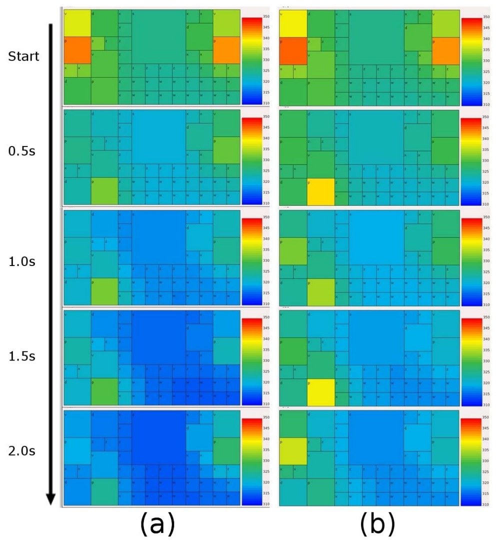

Fig. 6. Run-time thermal maps: (a) adaptive; (b) migration [6].

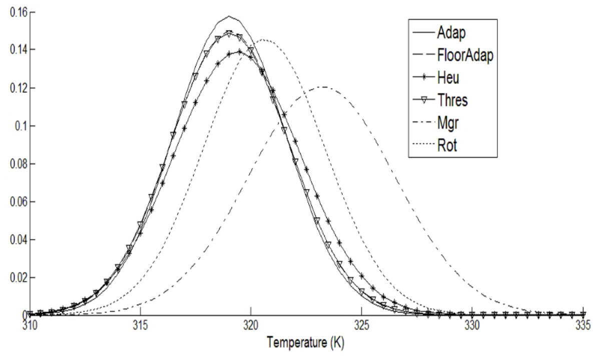

Fig. 7. Normalized statistical distributions.

Another interesting quality factor in the development of task migration techniques is the number of migrations per unit. As has been previously discussed, task migration policies introduce a performance overhead due to the time required for the memory allocation, as well as an energy waste. This impact can be characterized by means of the number of effective migrations per time unit. Figure 8 shows the number of migrations per time unit for all the policies considered in our study. As can be seen, our proposed policies not only achieve similar results to the threshold technique [5] in terms of mean temperature and sigma of the thermal distribution, but they also decrease the impact on performance by a 40% because less task migrations are required. Table III summarizes the performance overhead imposed by every task migration technique, where the minimum impact of our proposed policies can be observed.

Finally, two factors with a very strong impact on the reliability of

TABLE III
PERFORMANCE OVERHEAD.

	Adap	FloorAdapt	Heu	Thres	Mgr	Rot
Overhead (%)	0.85	0.52	0.85	1.2	0.93	2.4

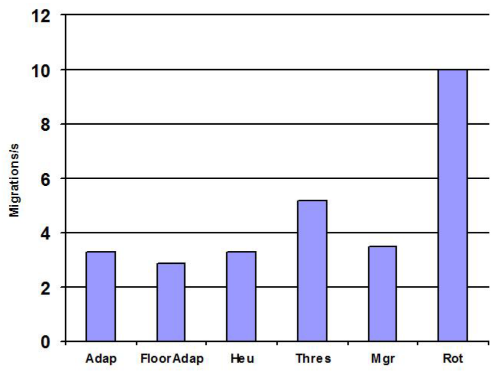

Fig. 8. Number of migrations per time unit.

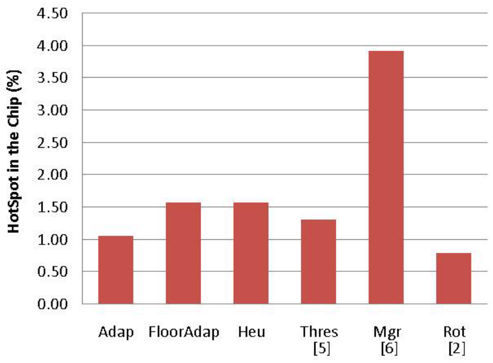

Fig. 9. Percentage of hot-spots.

the system have been evaluated: the percentage of hot spots in the chip area, and the thermal cycles. Both metrics have been calculated assuming that a hot spot in our set-up is represented by a temperature value over 338 K. Figure 9 shows the percentage of hot spots in the chip area, averaged along the execution of the benchmark, and for every migration policy. As can be seen, our Adaptive policy behaves better than the traditional approaches, only outperformed by the Rotation policy which, on the contrary, has a strong impact on performance. The percentage of hot-spots is reduced to 1% and, therefore, the probability of system failure is minimized.

Figure 10 shows the thermal cycles for the same system configuration and task migration policies. As can be seen, our proposed approaches are able to reduce the thermal cycles to a minimum, showing better results than the traditional approaches (25% better than [5] and up to 4× less thermal cycles than [6] and [2]); and, moreover, with the smallest performance overhead (less than 0.9% impact on execution time).

VI. Conclusions

In this paper, we have investigated and proposed OS-level task migration policies for thermal management in embedded multiprocessor systems. We have showed that the proposed techniques achieve low and balanced temperatures profiles, diminishing the percentage of hot spots, thermal cycles, and thermal gradients. As compared with traditional techniques, our policies incorporate the floorplan information in the OS, dynamically adapt the migration to the thermal profile of the application, and improve the thermal behavior of the chip with a negligible performance overhead.

References

[1] Semenov, O. e. A. (2006) Impact of self-heating effect on long-term reliability and performance degradation in CMOS circuits. *IEEE Transactions on Device and Materials Reliability*, **6**, 17–27.

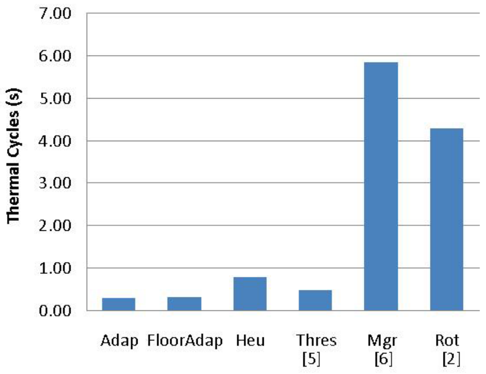

Fig. 10. Thermal cycles.

[2] Chaparro, P. e. A. (2007) Understanding the thermal implications of multi-core architectures. *IEEE Transactions on Parallel and Distributed Systems*, **18**, 1055–1065.

[3] Carta, S., Acquaviva, A., Del Valle, P. G., Atienza, D., De Micheli, G., Rincon, F., Benini, L., and Mendias, J. M. (2007) Multi-processor operating system emulation framework with thermal feedback for systems-on-chip. *Proceedings of the 17th ACM GLS on VLSI*, pp. 311–316.

[4] Atienza, D., Del Valle, P. G., Paci, G., Poletti, F., Benini, L., Micheli, G. D., Mendias, J. M., and Hermida, R. (2007) HW-SW emulation framework for temperature-aware design in MPSoCs. *ACM Trans. Des. Autom. Electron. Syst.*, **12**, 1–26.

[5] Mulas, F., Pittau, M., Buttu, M., Carta, S., Acquaviva, A., Benini, L., and Atienza, D. (2008) Thermal balancing policy for streaming computing on multiprocessor architectures. *Proceedings on DATE*, pp. 734–739.

[6] Gomaa, M., Powell, M. D., and Vijaykumar, T. N. (2004) Heat-and-run: leveraging SMT and CMP to manage power density through the operating system. *SIGOPS Oper. Syst. Rev.*, **38**, 260–270.

[7] Suen, T. T. Y. and Wong, J. S. K. (1992) Efficient task migration algorithm for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, **3**, 488–499.

[8] Chang, H. W. D. and Oldham, W. J. B. (1995, pages = 1301-1315,) Dynamic task allocation models for large distributed computing systems. *IEEE Transactions on Parallel Distributed computing Systems*, **6**.

[9] Barcelos, D., Brião, E. W., and Wagner, F. R. (2007) A hybrid memory organization to enhance task migration and dynamic task allocation in NoC-based MPSoCs. *Proceedings of the 20th annual conference on Integrated circuits and systems design*, pp. 282–287.

[10] Donald, J. and Martonosi, M. (2006) Techniques for multicore thermal management: Classification and new exploration. *Proceedings of the 33rd international symposium on Computer Architecture*, pp. 78–88.

[11] Yang, J., Zhou, X., Chrobak, M., Zhang, Y., and Jin, L. (2008) Dynamic thermal management through task scheduling. *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and software*, pp. 191–201.

[12] Paci, G., Marchal, P., Poletti, F., and Benini, L. (2006) Exploring temperature-aware design in low-power MPSoCs. *Proceedings of the DATE*, March, pp. 1–6.

[13] Flautner, K. and Mudge, T. (2002) Vertigo: automatic performance-setting for Linux. *SIGOPS Oper. Syst. Rev.*, **36**, 105–116.

[14] Huang, W., Stant, M. R., Sankaranarayanan, K., Ribando, R. J., and Skadron, K. (2008) Many-core design from a thermal perspective. *Proceedings of the 45th annual DAC*, pp. 746–749.

[15] Skadron, K., Stan, M. R., Sankaranarayanan, K., Huang, W., Velusamy, S., and Tarjan, D. (2004) Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, **1**, 94–125.