Proyecto Fin de Máster en Ingeniería de Computadores

"Máster en Investigación en Informática, Facultad de Informática, Universidad Complutense de Madrid"

Adaptive task-migration policies for thermal optimization in MPSoCs

Autor: **David Cuesta Gómez**

Directores: **José Ignacio Hidalgo Pérez**

José Luis Ayala Rodrigo

Curso 2008-2009

Departamento de Arquitectura de Computadores y Automática.

Facultad de Informática

*El/la abajo firmante, matriculado/a en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "***Adaptive task-migration policies for thermal optimization in MPSoCs***", realizado durante el curso académico 2008-2009 bajo la dirección de* **José Ignacio Hidalgo y José Luis Ayala** *en el Departamento de* **Arquitectura de Computadores y Automática**, *y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.*

Firmado:

David Cuesta Gómez

A mi hermano, padres

tíos, primos y abuelos.

A Belén.

Agradecimientos

Gracias a los que han transmitido sus conocimientos para hacer posible la realización de este proyecto.

A mi familia y amigos que siempre me han apoyado.

Al Politecnico di Torino por haberme permitido la oportunidad aprender de ellos y sobre todo por darme su amistad.

A Iñaki, José Luis y David, que desde el primer momento me abrieron las puertas y me acogieron como uno más.

A Dios porque me gusta tenerlo presente.

Index

Chapter III:

Chapter IV:

Resumen

En los circuitos tecnológicos diseñados con tecnologías de fabricación por debajo de la micra, las altas temperaturas provocan fallos críticos en la fiabilidad, el temporizado, los costes de refrigeración y la potencia de pérdidas. Hasta el momento se han propuesto varias técnicas de migración de tareas para manejar eficientemente la distribución térmica en los sistemas multiprocesador, pero con un alto coste en cuanto a la eficiencia del sistema.

Aunque las técnicas tradicionales se han centrado en reducir la temperatura media del chip, no han considerado los efectos que los gradientes térmicos tienen en la fiabilidad del sistema.

En este trabajo, se exploran los beneficios de las técnicas de migración de tareas basadas en la temperatura en sistemas empotrados multiprocesador. En particular se proponen algunas políticas que son capaces de reducir la temperatura media del chip y los gradientes térmicos con un impacto en el rendimiento prácticamente despreciable.

Con nuestras técnicas, la aparición de puntos calientes (hot spots) y variaciones térmicas se ven drásticamente reducidos con respecto a otras propuestas, con lo que la fiabilidad del sistema se mejora significativamente cuando la comparamos con las técnicas de migración de tareas tradicionales

Abstract

In deep submicron circuits, high temperatures have created critical issues in reliability, timing, performance, cooling 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 variations are decreased, and the reliability of the system is significantly improved when compared to traditional task migration techniques.

Palabras Clave

Temperatura, migración de tareas, algoritmo adaptativo, sistemas multiprocesador, fiabilidad, preocupación térmica, colocación de floorplan.

Keywords

Temperature, task migration, adaptive algorithm, multi-processor system, reliability, thermal aware, floorplan placement.

Introduction

In this section a global view of the problem is presented. The state of the art is also revised doing a comparison between the proposed traditional techniques and ours.

Structure of the Memo

This work has four main parts, each one described as follows:

I. In this first part a slight description of the whole work is presented, introducing the formalism and establishing the formalism and lexicon that will be common to the whole work.
II. In this second part of the memo, a more intensive description of the work done will be presented. The work done is divided in three main blocks:
    a. Hardware
        i. Understand the FPGA design and configuration when collecting and sending data to the mainframe.
    b. Software
        i. Understand and edit operative system allocated in the FPGA cores, which will have the task migration policies included in it.
        ii. Understand the two dimension thermal model, which will give us the temperature of each cell in the floorplan thanks to the information provided by the emulation FPGA platform.
    c. Analyze
        i. Analysing the results given by the thermal model using the developed graphic tool.
        ii. Offline extraction of thermal statistics, which will help us to know how good the thermal policy is.
III. In this third block, discussed results obtained are presented.

In this last part conclusions and comments about the work done are described. Also possible future work is proposed, basing it in the work done.

Introduction

Mobile System-on-Chip (SoC) devices count nowadays with multiple processors in their implementation to fulfil the demanding performance requirements. Besides that, these mobile Multi-Processor System-on-Chip (MPSoC) platforms will represent a significant portion of the media market in a near future [1]. However, the increase in power density (due to the integration of many active components per area unit), and the capability to control their thermal behaviour are two of the key factors that limit the performance in MPSoC architectures [2].

Recent works have demonstrated that large temperature variations could cause low reliability and they also negatively impact on leakage current. Temperatures over a threshold in localized areas of the chip (hot spots) as can be seen in Figure 1 can produce timing delay variations, transient reduction in overall system performance or even permanent damages in the devices [3].

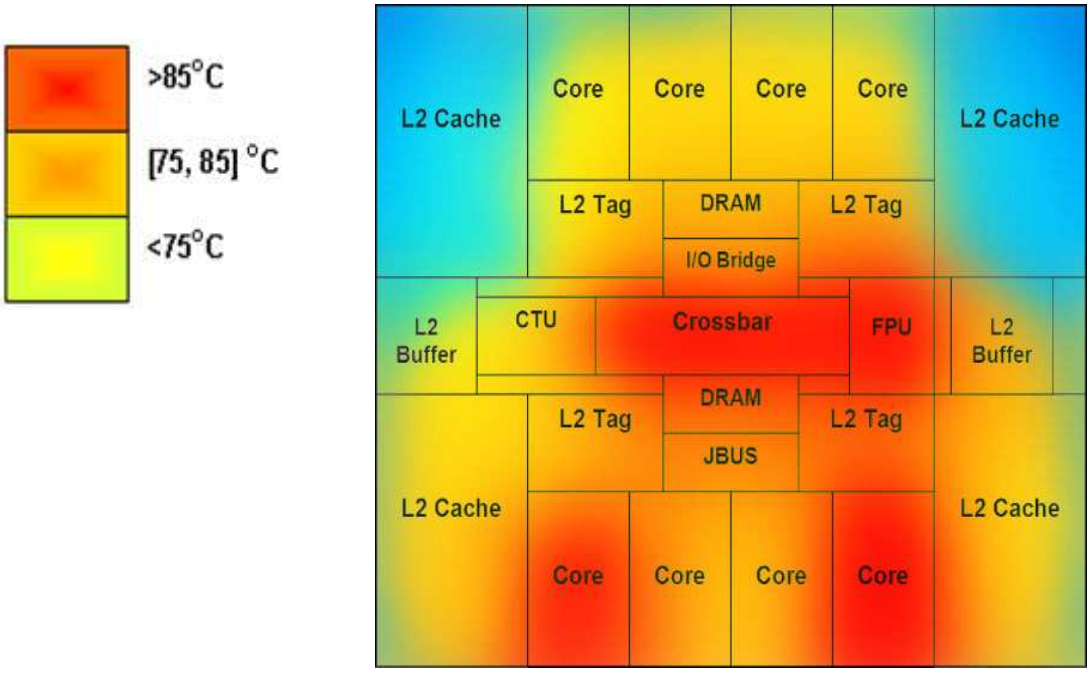


Figure 1: Hotspots in a Niagara broadband processor

Packaging, heat sinks and cooling solutions can be proposed to minimize the impact of temperature in performance, but their high cost complicates the general adoption of such techniques [4]. Moreover, the reliability factors do not only depend on the average temperature of the chip, but also the spatial and temporal

variations have a strong influence in phenomena like electromigration, negative bias temperature instability or thermal cycles [5].

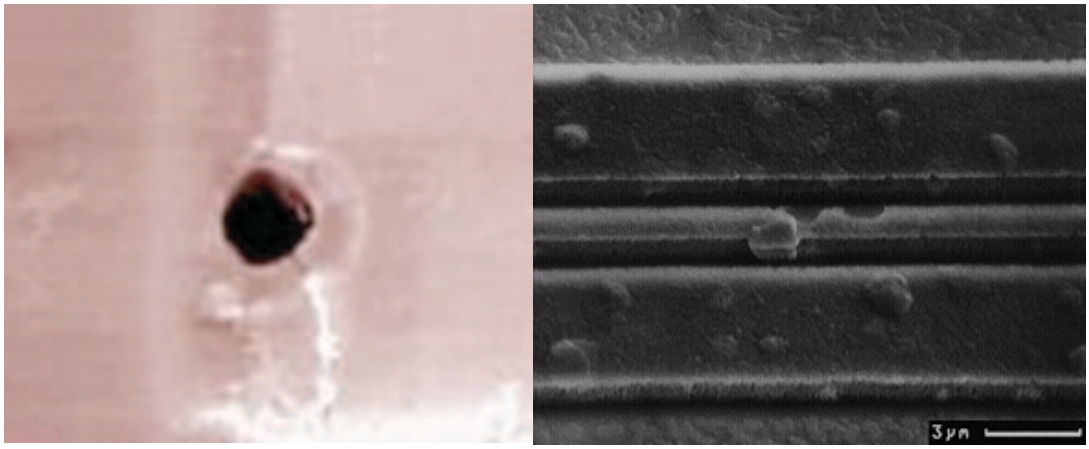


Figure 2: Electromigration seen with a microscope

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 efficiently the thermal profile in high performance and embedded multi-processor systems [6, 7].

While traditional dynamic thermal management (DTM) techniques have been devoted to decrease the peak and average temperature of the chip, they have not considered the spatial and temporal gradients that determine the meantime-to-failure of the devices.

The evaluation of thermal policies and task migration techniques is a very computational intensive analysis. Thermal simulation of complex MPSoCs, where the exploration of the interaction between the hardware architecture and the software layer that performs the task migration is also crucial, can take an unaffordable time.

Thus, in order to explore the HW/SW interaction, FPGA based emulators have been developed [8, 9]. These platforms provide the required accuracy and flexibility in the thermal hardware-software analysis without impacting the emulation time. Moreover, these infrastructures include the software layers and libraries needed to support the task migration and dynamic voltage and frequency scaling (DVFS) policies, namely, the multi-processor operating system (MPOS), the

middleware and the communication library. The experimental work carried out in this work is developed for an FPGA-based emulation platform that speeds up the simulation time and provides high flexibility in thermal analysis.

Thus, this work 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. These policies are embedded in a real MPOS that applies them on a real life MPSoC, and our results show that they reduce the impact on the system performance to a minimum as compared to previous published approaches [10, 6, 7, 11]. The specific contributions of our work are the followings:

- Three task migration policies are proposed (one heuristic, one adaptive technique and a third floorplan-aware adaptive policy), capable of optimizing the thermal diffusion in MPSoCs.
- The proposed policies minimize the peak temperature and the thermal gradients, with a reduced performance overhead.
- The reliability of the system is improved by the minimization of the number of hot spots and the thermal cycles.
- The experimental work has been carried out in a realistic emulation platform, and the task migration policies have been implemented in a real-life uClinux-based [27] multiprocessor operating system.

State of the Art

Load balancing techniques have been deeply studied for general purpose parallel computers in the last decade [12,13]. However, embedded systems and MPSoCs impose constraints, as the low-cost packaging and the portability, that make necessary to develop new techniques. Nollet et al. [14] proposed a reuse technique that uses the debug registers of the processor to get the system workload information. Therefore, the initial overhead of a heterogeneous MPSoC task migration is diminished by considering these hardware devices which are not always available in current architectures.

Bertozzi et al. [15] presented an approach that dealt with MPSoCs task migration. They proposed a strategy where the user is responsible for setting the

possible migration points in the application code. The architecture used in this work was composed by one master and an arbitrary number of slaves cores. Even though this paper shows interesting results for such specific architecture, our work deals with amore general system where the task migration is dynamically performed.

Götz et al. [16] present a design flow for dynamic relocation of hybrid tasks. These tasks may be executed either in hardware or software and are represented through a *state transition graph*, where each state is known as computation block and stands for a given task operation. Our work outperforms this approach by a careful selection of the threshold mechanism that decides the migration point while preserving system performance. Barcelos et al. [17] 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. Barcelos' work is extended by Brião et al. [18] who takes into account the task migration overhead in a dynamic environment and discusses its impacts in terms of energy, performance and real-time constraints for MPSoCs based on Network on Chips (NoCs). Following this line, our work considers the impact of task migration and minimizes this factor to optimize both performance and energy dissipation.

In the area of temperature optimization, several approaches have been proposed to reduce the peak temperature through task-migration techniques. Donald et al. [7] introduced several thermal management policies such as DVFS and thread migration based on current temperature, but their work do not consider the thermal history of the cores. This information gives meaningful information about the future behaviour of the system and can be exploited to improve the results of the migration. The work by Puschini et al. [19] also manages dynamically the voltage and frequency assignment of each core based on game theory. This scheme is aimed to target a scalable mechanism with many cores. However, the DVFS as a thermal optimization technique is limited by the implementation and its impact on performance.

On the other hand, Powell et al. [20] described techniques that, using the information provided by performance hardware counters, tried to balance the temperature by thread migration. However, it is considered that performance counters do not represent accurately the thermal profile. In [21], Yang et al. showed an execution ordering approach that swaps hot and cool threads in cores to

control the temperature. This idea can only be applied once the application has been profiled to obtain the thermal information per thread, which means sometime an unaffordable time.

Finally, a recent work by Yeo et al. [22] presented a temperature-aware scheduler based on thermal grouping of the applications using a K-means clustering. This work provided interesting results but requires a very complex analysis phase, which grows largely in complexity with the number of considered cores.

Our work outperforms previous approaches with the provision of three task migration techniques that optimize the thermal profile of MPSoCs with very low performance overhead. Moreover, our techniques are able to minimize the risk of system failure by the minimization of temperature driven reliability factors, and can be applied to complex systems with a large number of integrated cores.

Objectives

Main objective of our work is to establish Operative System policies which will be able to manage task loading among the cores to keep a homogeneous thermal distribution in the whole chip through task migration policies.

To make this work several steps must be done in order to follow a normal project execution. These steps will be considered as previous work or partial objectives, and they will be:

- Previous study of the related work
  - Migration policies
  - Thermal aware methods
  - Simulation / emulation methods
- Analysis of the experimental environment
  - Emulation platform
  - Linux distribution
  - Operative system characteristics.
- Design and test of some simple task migration policies
- Design of a graphic user interface to analyze results.
- Qualitative and quantitative study of the results, creating a thermal metric to evaluate the goodness of a policy, evaluating:
  - Mean temperature
  - Maximum temperature
  - Temperature gradient
- Design of optimized policies.
- Test and validation of the proposed policies.
- Analysis of the results.

Chapter I

Thermal Model and Emulation Platform

Thermal Model

The thermal model used in this work is thought a $R_{thermal} - C_{thermal}$ model in which thermal behaviour will be modelled as an electric circuit. We will see an example presented by G. Paci et al. [23].

It shows a typical low power multiprocessor on chip (LP-MPSoc). This system is showed in Figure 3, where a 16 ARM7 cores and 16 32KB shared memories are presented.

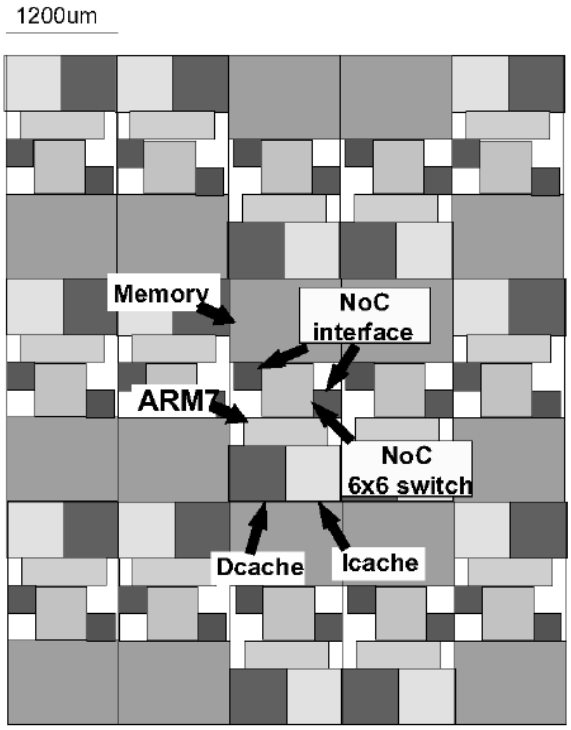


Figure 3: Floorplan of a LP-MPSoC

Each core, is attached to a local 8KB data cache and to a 8KB instruction cache. The memories and the cores are connected using a XPipes Network-on-Chip.

To know the power consumption and then the temperature of all the blocks inside our chip a simulation environment must be created. This environment must estimate the power consumption of the components of the chip. This power will depend on the workload in the processors and memories. In this model leakage power is not contemplated because the system is implanted in an embedded system so to assure a sufficient battery-life time the leakage must be reduced to negligible values. So power estimation values are provided by this work [23]. These values are shown in Table 1.

	Max power @100MHz (mW)	Max. Power density (W/mm²)
ARM7	5.5	0.03
DCache	43	0.012
ICache	11	0.03
Shared memory	15	0.02

Table 1: Power for the most important components of a LP-MPSoC in CMOS technology

After having a power consumption estimation an horizontal heat flow model is needed. The heat flow is permitted by the package, and this package is the one in charge of dissipating this heat to the environment. This model considers every surface as adiabatic but the die package. In Figure 4 the heat flow parts can be seen.

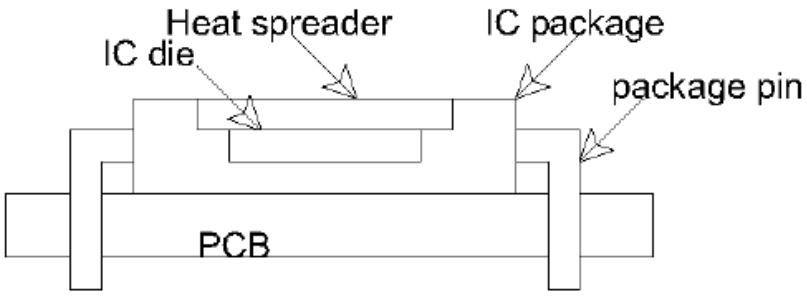


Figure 4: Chip package solution

Equivalent RC Thermal Model

This model as we said before, is based in the well known analogy between electrical circuits and thermal models. The silicon is decomposed in elementary cells which have a cubic shape. This work is also done with the heat spreader of the chip. To solve each cell, a RC computation model is needed. Each cell is associated with a capacitance and five thermal resistances which are used for modelling both, horizontal thermal spreading and vertical thermal behaviour. This can be seen in Figure 5. The values for the resistances and the capacitance are calculated taking into account the values for silicon thermal conductivity and cupper thermal conductivity.

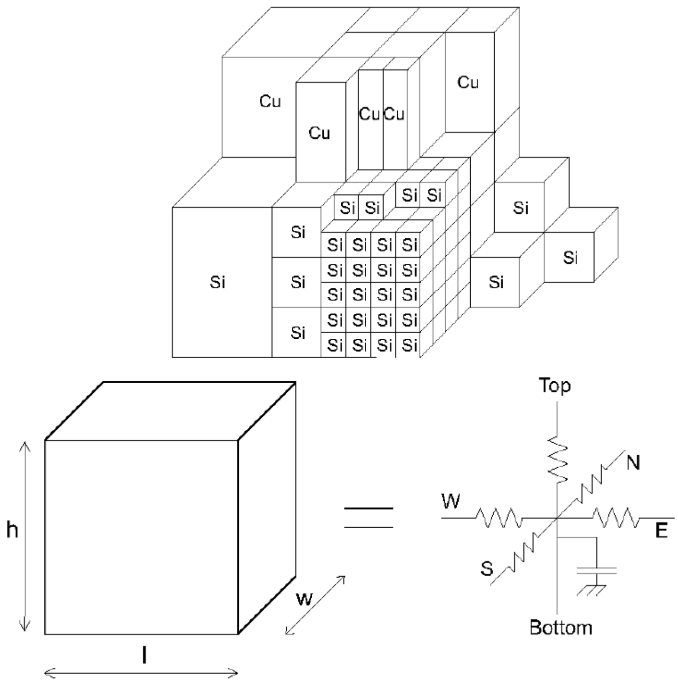


Figure 5: Cells division and equivalent RC circuit of a cell

We will ise the following description: $k_{th}^{si/cu}$ is the thermal conductivity and $c_{th}^{si/cu}$ is the capacitance. Both are considered per unit volume. The cell size units are represented by *l*,*w* and h, as shown in the Figure. The following equations show in detail how the values are calculated:

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

Equation 1: Admittance for a cell

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

Equation 2: Capacitance value for a cell

With these values calculated for each cell the thermal model solver is executed. It consists in solving several differential equations in an iterative way. Every cell in the floorplan has an associated equation which describes its iteration with its neighbours. This equation is written as follows:

$$\frac{C_{cell}}{\Delta t}\left[T_{cell}(i+1) - T_{cell}(i)\right] = Gn1\left[T_{n1}(i) - T_{cell}(i)\right] + \ldots +$$
$$+ Gnx\left[T_{nx}(i) - T_{cell}(i)\right] + \ldots + G_{nm}\left[T_{nm}(i) - T_{cell}(i)\right] \mid S_{cell}(i)$$

Equation 3: Differential equation for each cell

Where:

G_{nx}: conductance between the n cell and neighbour x

n: number of cell

x: $1 \leq x \leq m$ is the position number of the neighbour cell

m: total number of neighbours cells

$T_{cell}(i)$: cell temperature at ith time step

$T_{nx}(i)$: temperature of the neighbour x of the cell n at ith time step

C_{cell}: cell capacitance

$S_{cell}(i)$: power burned at ith time step

Δt: time between two time step.

It must be noted that silicon thermal conductivity is not linear and it depends on the temperature. To ease the computation, we have approximated it by its first order Taylor series.

Technologic values for our parameters are depicted in

Table 2.

Silicon thermal conductivity	$150 \cdot (300/T)^{4/3}$ W/mK
Silicon specific heat	1.628e-12 J/µm^{3}K
Silicon thickness	350 µm
Copper thermal conductivity	400 W/mK
Copper specific heat	3.55e-12 J/µm^{3}K
Copper thickness	1000 µm
Package-to-air conductivity	20 K/W (in low power)

Table 2: Thermal properties

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.

Figure 6: Virtex II Pro, VP30

In this work, we have used a complete FPGA-based estimation framework, implemented in a Virtex II pro VP30 (Figure 6) and inspired by the work in [9]. Figure 7 shows a schematic view of this emulation platform detailing a single core

system. As can be seen, within 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 matches) by mean of hardware sniffers. The work in [9] has also been extended to allow the characterization of a system with three working cores and one arbiter as the one considered [8].

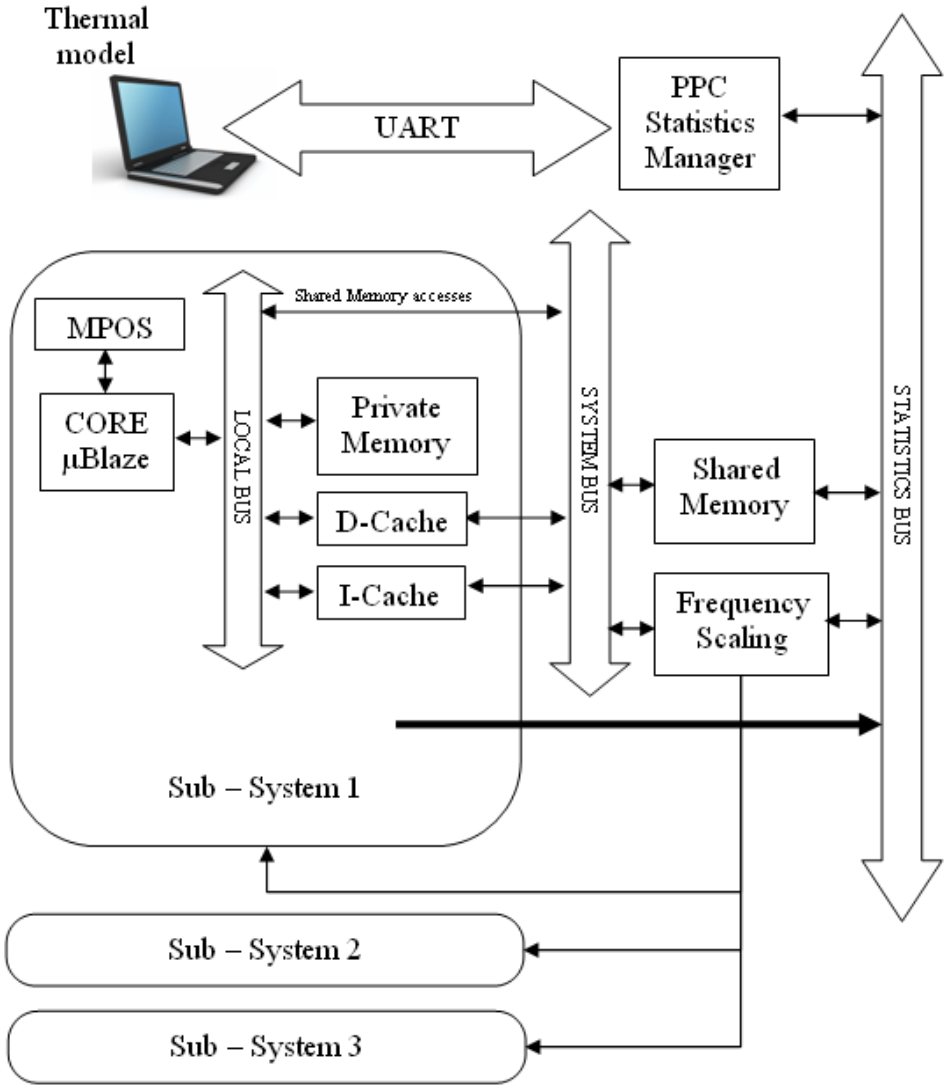


Figure 7: Description of the emulation platform

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 modelled as simple $R_{thermal}C_{thermal}$ circuits as said before.

The resolution of the linear equations created by the RC grid provides the evolution in time of the temperature of the system [23].

The emulated architecture is a homogeneous multiprocessor system with three 32-bit RISC cores and the PowerPC. These processors 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 [26]. 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.

There are two techniques to migrate tasks among cores: [10]

- Task recreation
- Task replication

The first one kills the process on the original processor, and then it recreates the same process from the scratch memory to the destination core. This strategy only works in those operative systems which support dynamic loading. In our case, our uClinux distribution is not prepared for that. This technique is based on the execution of fork - exec system calls, which take care of allocating the memory space required for the incoming task. In order to support task recreation in a system without MMU, extra hardware is required to prevent the generation of wrong reference of pointers, since the starting address of the memory can change during the execution. This is a great withdraw in our system because our core does not support this extra hardware called PIC.

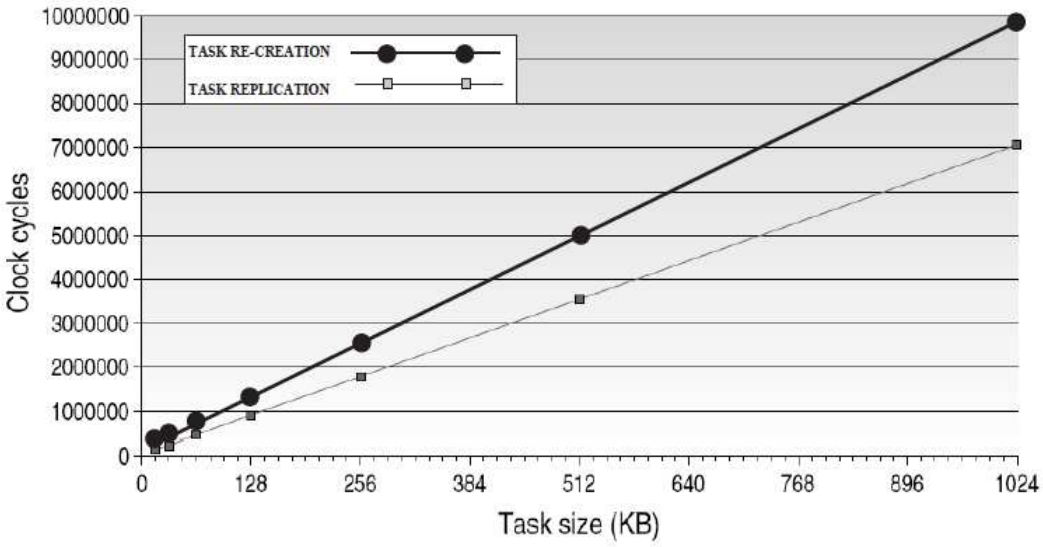


Figure 8: Migration cost as a function of task size for task replication

Because this limitation, and also because of its migration speed we implemented in our system the second technique for task migration, *task replication.*

With this technique only one processor at a time can run one replica of the task. While in one processor the task is executed normally, in the other ones, it is in a queue of suspended tasks. This means that a memory area is reserved for each replica in the local memory, while kernel-level task-related information is allocated by each OS in the Process Control Block. Therefore, task replication is suitable for deeply embedded operating systems without dynamic loading because the absolute memory address space does not change upon migration, since it can be statically allocated at compile time. In fact, even if this technique leads to a waste of memory for our tasks, it has the advantage of being faster, since it cuts down on memory allocation time with respect to a task recreation.

A quantification of the memory overhead due to task replication is shown in Figure *8*. In this figure, the costs are shown in terms of processor cycles needed to perform a migration as a function of the task size. In both techniques the major part of the overhead is due to the data transferred through the shared memory. For task recreation technique, another overhead must be taken into account. This overhead is produced due to the time required to reload the program code from the file system; thus the offset that can be seen in the figure as the gap between the two curves.

Moreover, the task recreation curve has a larger slope because the larger the memory transfer is, the more it takes to re load the program, besides it leads to an increasing contention on the bus. Hence, the contribution on the execution time increases as file size increases in comparison to the task replication case.

The task migration takes place only at predefined checkpoints chosen by the programmer. A master daemon runs in one of the cores, which dispatches then tasks to the processors.

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. In summary, the complete emulation platform is composed of the following abstraction layers:

- **Application layer**: built as a set of independent tasks found in every processor of the system. The tasks are executed under the OS demand.
- **OS/Middleware layer**: controls the task migration and the communication and synchronization of the cores through the shared memory.
- **HW layer**: composed of three core-subsystems and a shared memory.

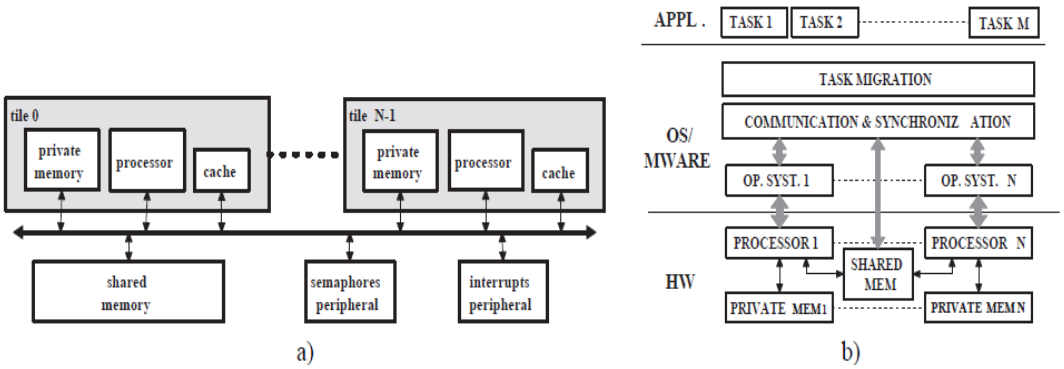


Figure 9: a) Target hardware architecture b) Scheme of the software abstraction layer

Finally, the emulation system has also been upgraded with a representation graphical tool for instrumentation purposes. This tool communicates with the thermal library and, in real time, provides a coloured thermal map of the emulated system.

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.

Graphic Interface

To communicate with the host PC, as we said before, the FPGA sends the information related to power consumption via serial port, so the host PC can know what the power consumption in each functional unit is.

This information is collected by the thermal model, which computes the temperature of each cell and sends cores and memories temperature back to the FPGA so the OS can manage the thermal policies.

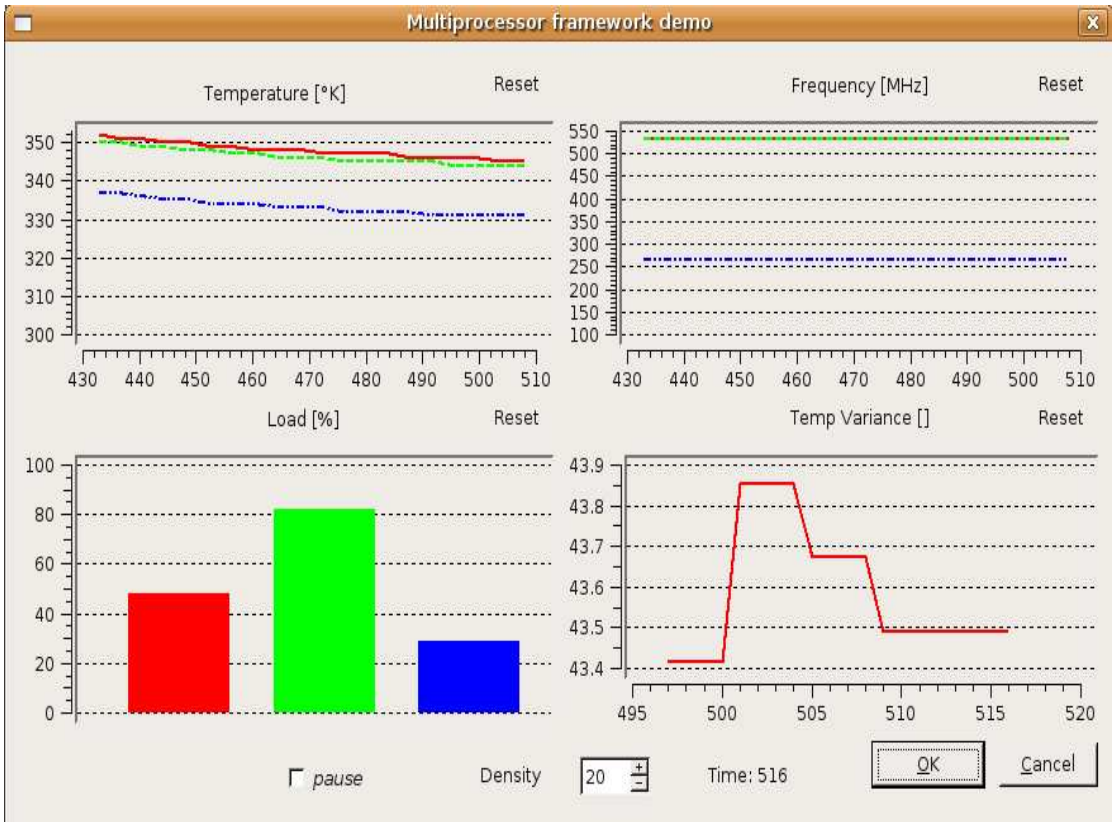


Figure 10: Platform graphic user interface

In the host PC the information is showed in a graphical interface, designed in order to understand what is going on with our emulated system.

This interface shows the temperature of each core, their workload, their frequency and finally it computes temperature deviation.

A snapshot of this interface is shown in Figure 10 where all these values can be seen.

Apart from this interface, 5 minicom terminals open at the same time, to get information and statistics of each core and the whole system in real time.

There is a minicom for each core. Another one shows the frequency and temperature information of the cores, and finally the last one shows the tasks queues identifiers.

Using this minicoms communication with the cores is allowed so we can change the DVFS policy or the migration policy we are using in real time.

A view of the minicoms is showed in Figure 11.

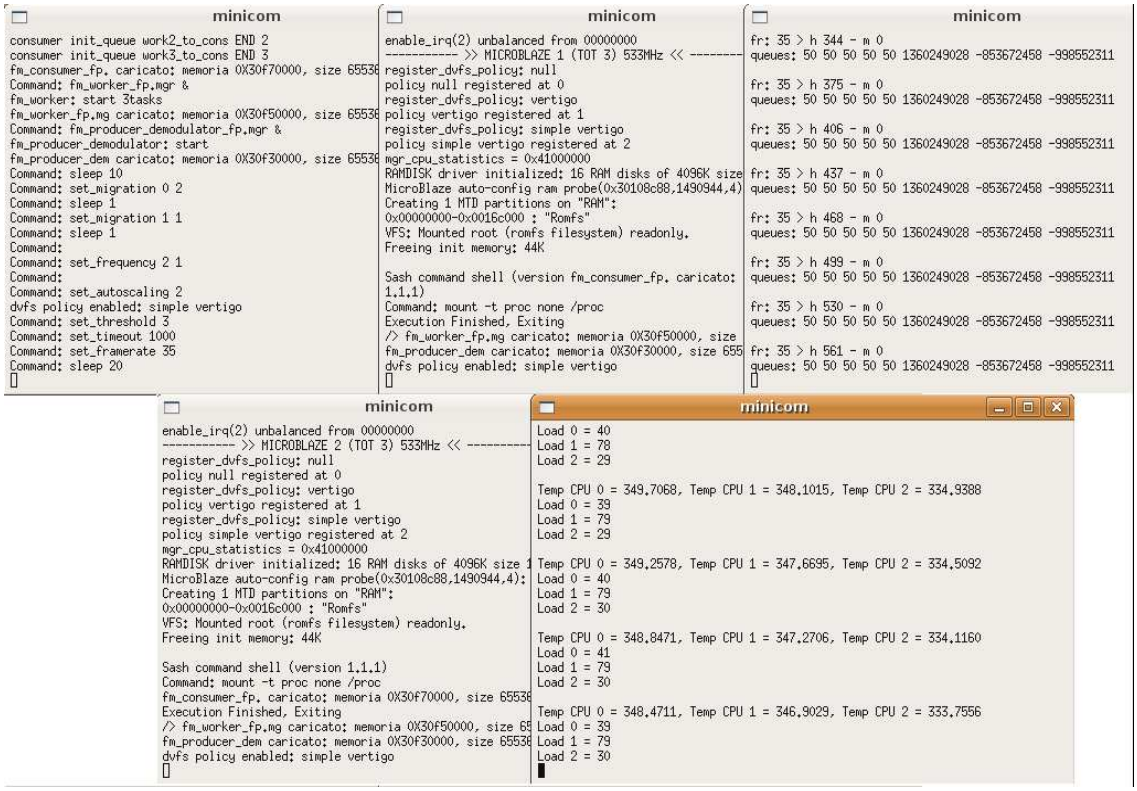


Figure 11: From high left to bottom right. Minicom Core 1, Minicom Core 2, Minicom Tasks queues, Minicom Core 3, Minicom temperatures and frequencies information

Using these graphical tools a rapid inspection of the system can be done. Getting information in real time is very important to know if the OS is managing the task migration policies correctly.

Combined with these tools another graphical tool was included to allow the user watching how the temperature evolves in real time. An offline study is also possible because temperature data are logged.

In Figure 12 a vision of this tool is showed. In it we can see how the blocks are painted in different colours according to their temperature.

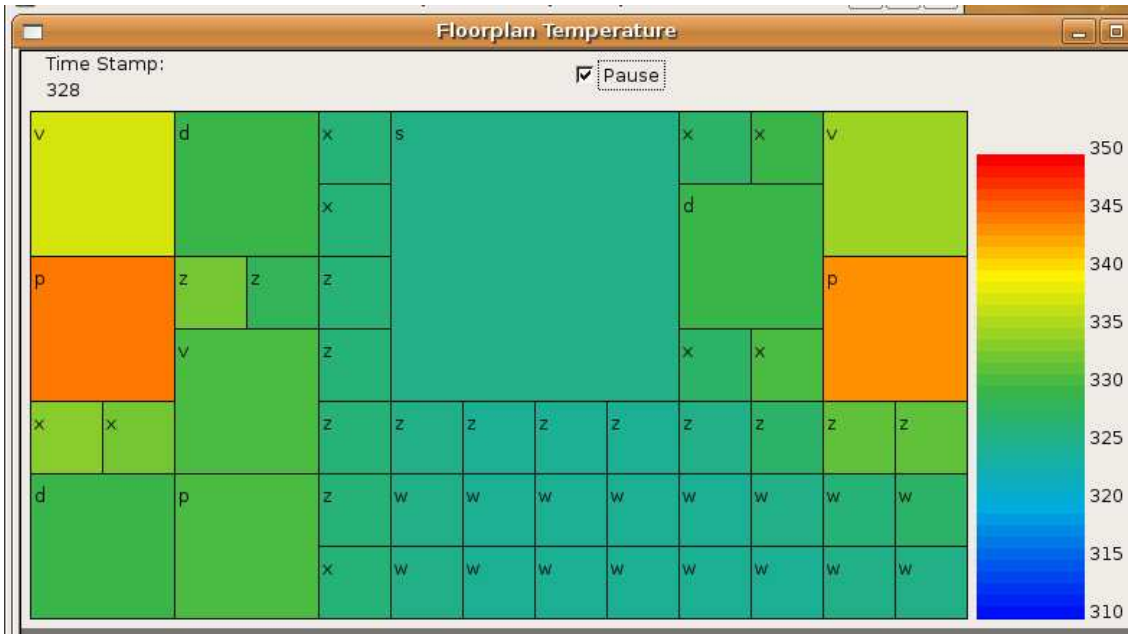


Figure 12: Floorplan graphic tool

This tool has also another advantage. It easy integrates new floorplans with a negligible cost in time. For example testing a new floorplan would only take 15 minutes. After this time qualitative and quantitave results could be extracted from the emulation and the statistic study of the most important parameters such as maximum temperature or mean temperature could be done. In Figure 13 we can see another example of a floorplan implemented in the system and executed showing cells temperature.

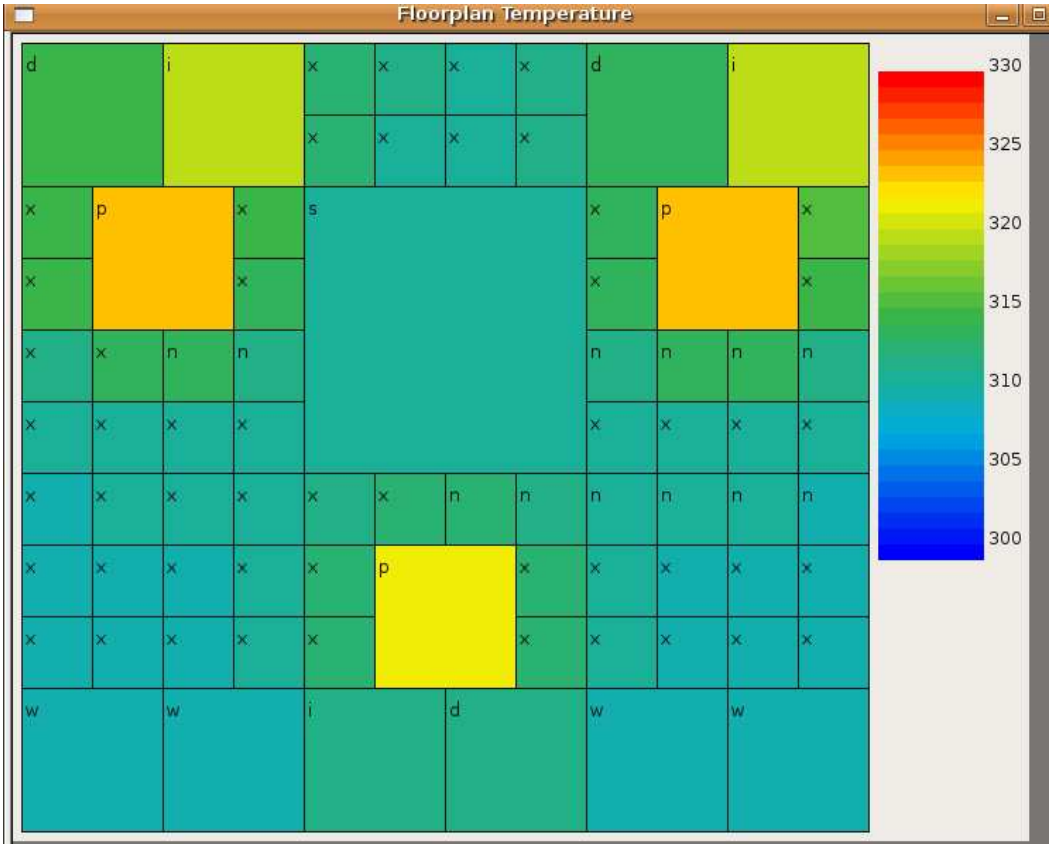


Figure 13: Extended flooplan example

Chapter II

Adaptive and floorplan aware policies for thermal balancing

As previously mentioned, the task migration policies we present in this work 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 [3]. 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 in the system follows the *vertigo* policy [24]. The application of the vertigo policy requires the previous characterization of the tasks 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 working load in the processors of the emulation platform and, consequently, obtain a homogeneous distribution of temperature in the system. Figure 14 presents a migration example. Three cores are running four tasks exhibiting different workload per processor. Workload in the processors is directly 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, as can be seen in Figure 14.

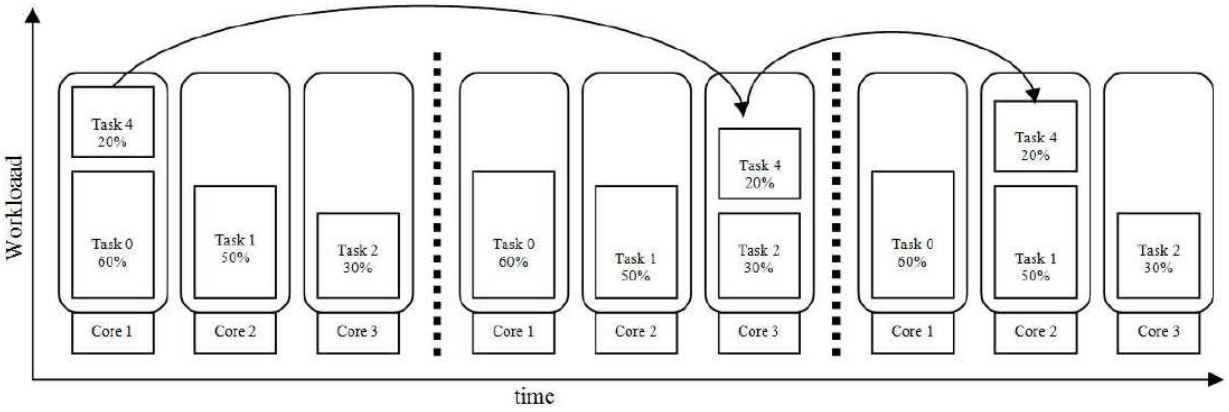


Figure 14: Task migration example

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 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 that we have been implemented in the considered emulation platform, and the policies we propose to specifically adapt the workload of the system depending on the state of the processors.

State of the Art Policies

Several migration policies have been proposed in the literature. Some were implemented in our system to compare them with our proposed policies.

We are going to vaguely explain what these policies consist on:

- **Enhanced Migration (Mgr)** 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 upgrade of the heat & run policy presented in [11] because it adds task migration. The implementation of this policy is based on the work by [10].
- **Task rotation (Rot)** [6], 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 [10], 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. It acts in both, hot and cold cores.

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 have been designed to 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 with a negligible computation cost. Figure 15 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 0.5% in performance.

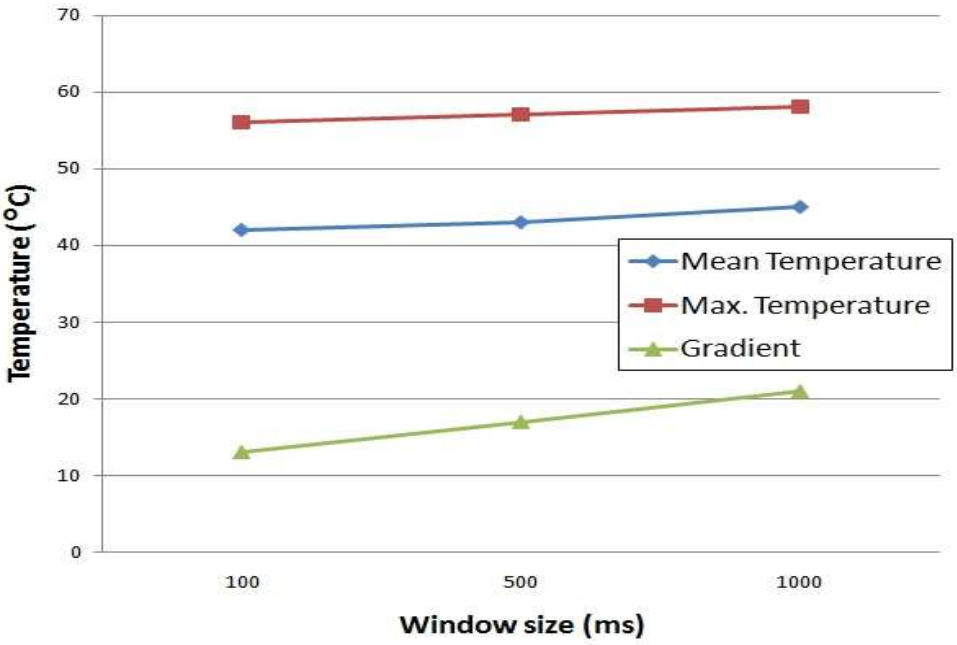


Figure 15: Impact of the time window

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.

Atomic Policy	Mean Temperature	Maximum 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

Table 3: Characterization of atomic policies

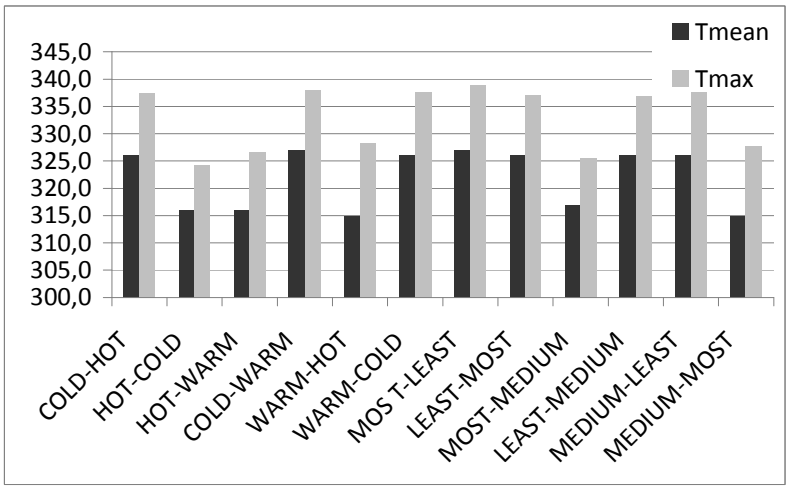


Figure 16: Mean and maximum temperatures for atomic policies characterization. Temperature polices and load policies

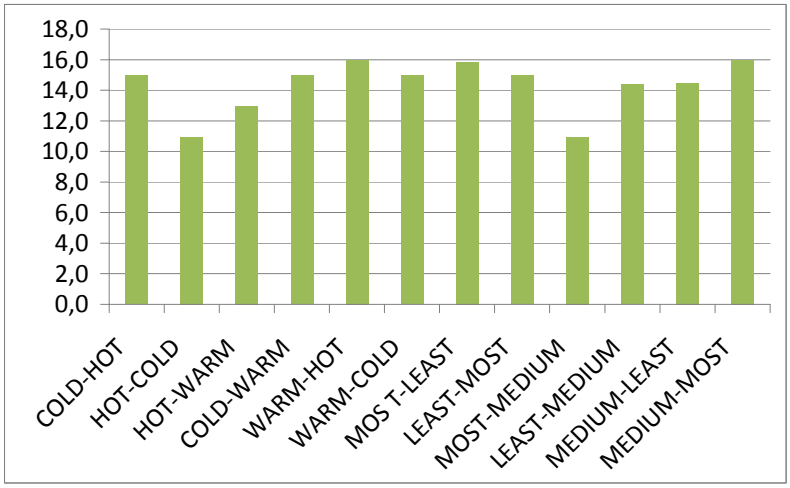


Figure 17: Thermal gradient for atomic policies characterization. Temperature polices and load policies

Table 3 shows a reduced sub-set of the atomic policies that have been considered and their classification after the statistic analysis. 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 for task migration has been set experimentally to 300 ms. Figure 15 shows the impact of the time window in the predefined metrics. A too small time window will affect the performance because of extra and unnecessary task migrations that present an overhead in the system functioning. On the contrary, a too big time window will create large temperature gradients, as shown in Figure 4, and will increase the probability of hot spots. Therefore, the time window has been selected as the largest with the minimum impact on the

temperature gradient. This selection is independent of the application run by the processors and only should be revisited in case of a new package.

Proposed Policies

Heuristic algorithm (Heu)

This algorithm is able to select efficiently among the atomic policies to 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 is collected at run-time during this time slot. At the end of the time window, we evaluate the collected 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 to solve the divergence of metrics between the current state and the desired one. Figure 18 shows the decision chart that explains the functioning of this heuristic.

In this Figure several parameters appear. *Deviation* is the difference between the preferred working value (which is 50ºC for the mean temperature, 70ºC for the peak temperature and 6ºC 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. *Factor* values are, 1 for mean temperature set, 1.5 for maximum temperature and 2 for the gradient.

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 decisor, an adaptive policy is introduced.

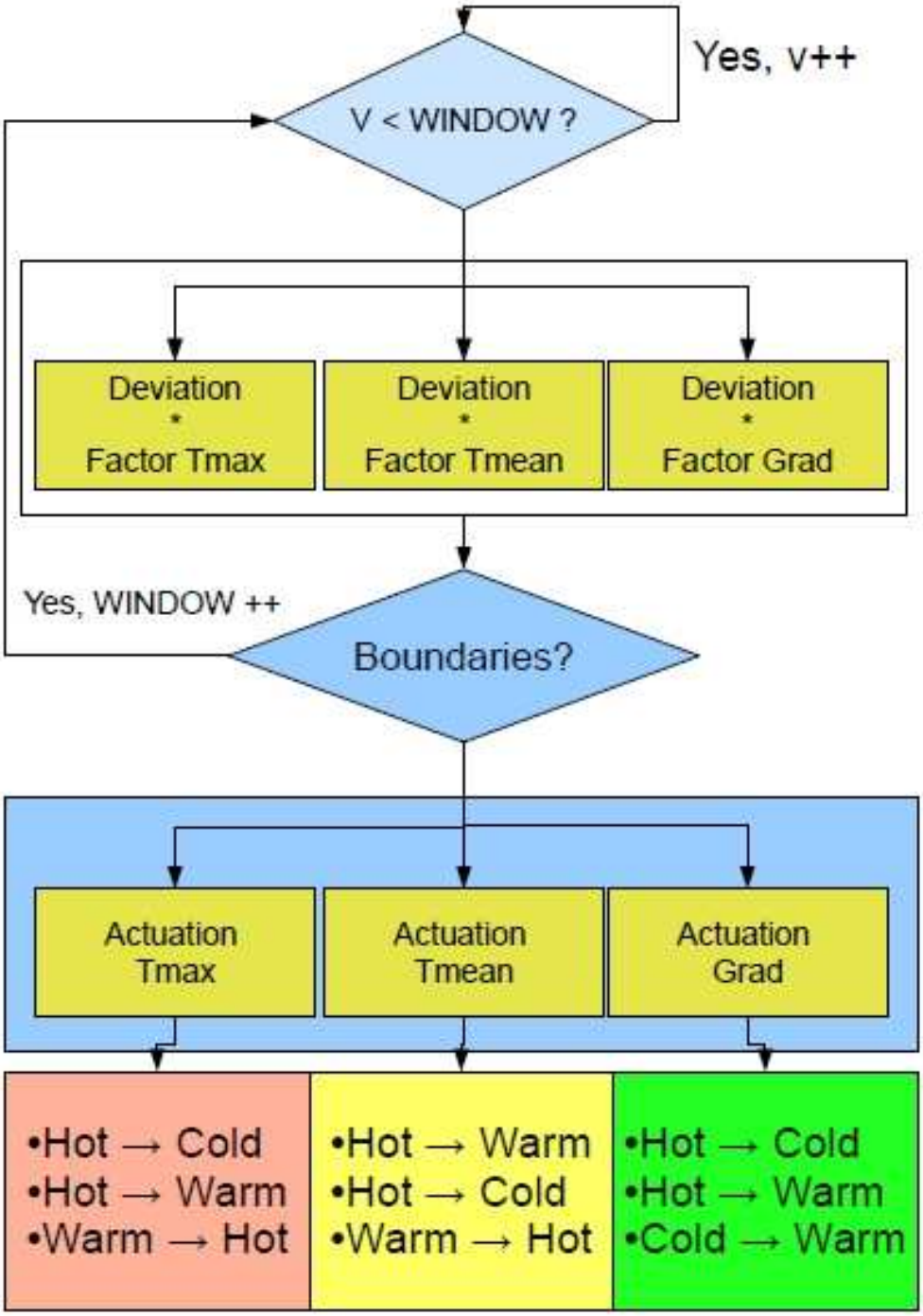


Figure 18: Heuristic algorithm decision chart

Adaptive Policy (Adapt)

This policy extends the work performed by the previous heuristic approach, collecting data at run-time and applying the atomic policies to achieve the optimum thermal state.

However, this policy adapts the selection of the atomic policy by means of the statistical information of the cores, which predicts the behaviour 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:

$$P_t = P_{t-1} + W$$
$$W_{init} = M_{pref} - M_{avg}$$
$$W = \begin{cases} \alpha_{inc}(T_{mean}, T_{peak}, T_{gradient}) \cdot W_{init}; & W_{init} > 0 \\ \alpha_{dec}(T_{mean}, T_{peak}, T_{gradient}) \cdot W_{init}; & W_{init} < 0 \end{cases}$$

Equation 4: Probabilities calculus

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} would also denote 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.

Finally, if a core trespasses the limit of 75ºC, it migrates all its tasks to the other processors in the system. In other words, we shutdown the core to avoid

heating it up too much and prevent reliability failures. Even if it is a safety measure, in our experiments it never happened.

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 sensed variables with the placement of the core to perform a more accurate selection of the destination core.

Floorplan-Aware Policy (FloorAdapt)

This policy considers the information about the floorplan. In this way, the OS is aware of the cores location 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:

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

Equation 5: Goodness of a processor to receive a task

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 [25]). 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 consider the proximity of the heat/sink and the thermal resistance of the joint.

Every time window, the thermal history of the processors is analyzed and updated to solve possible hot spots, critical thermal gradients, or values over the safe peak temperature. 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 behaviour due to the proximity to heat sinks or heat sources which dissipate the temperature. In our floorplan design shown in Figure 19, **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.

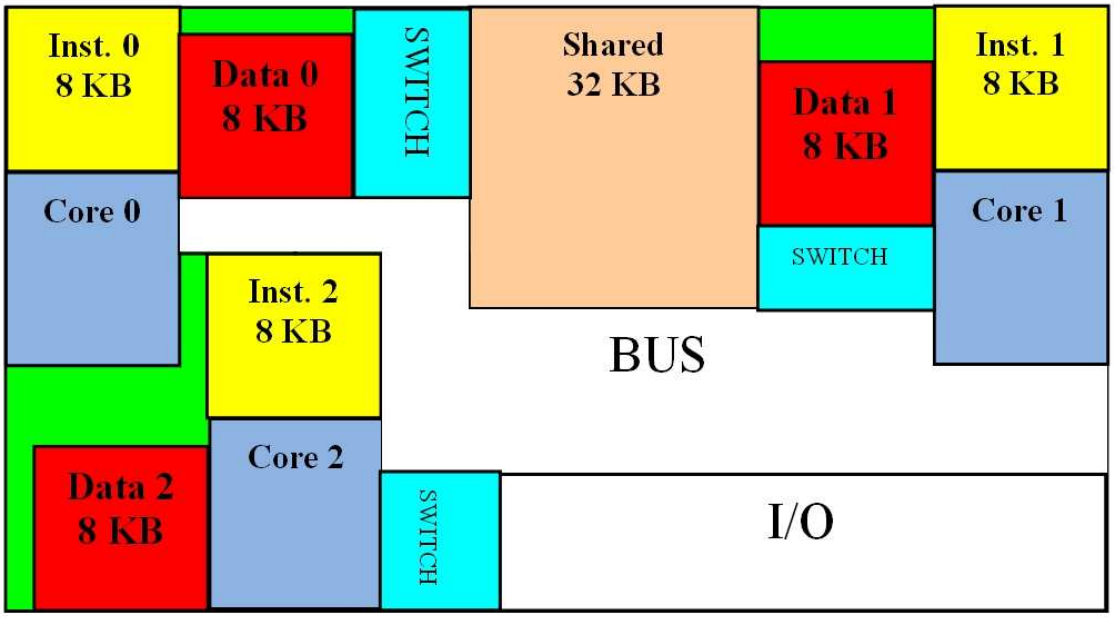


Figure 19: Floorplan design

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

Chapter III
Experimental Results

The experimental work has been conducted with the emulation platform described in previous sections, 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 and statistics collector. The benchmark selected for the analysis is a real-life streaming application capable of loading the three cores.

The experiments have been run considering a special package derived from real-life streaming SoCs [2] for mobile embedded devices. In a target system as the one resembled, 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:

- 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.
- Mean temperature of the chip: calculated as the arithmetic mean of the processor and memory temperatures in the chip. This metric relates the temperature of the devices to the energy consumption and cooling necessities.
- 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 before.

Description of the Application

The software that is executed by the platform is a Software FM Defined Radio (SDR) which is a perfect example of streaming application. This application is composed of several tasks that can be perfectly 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.

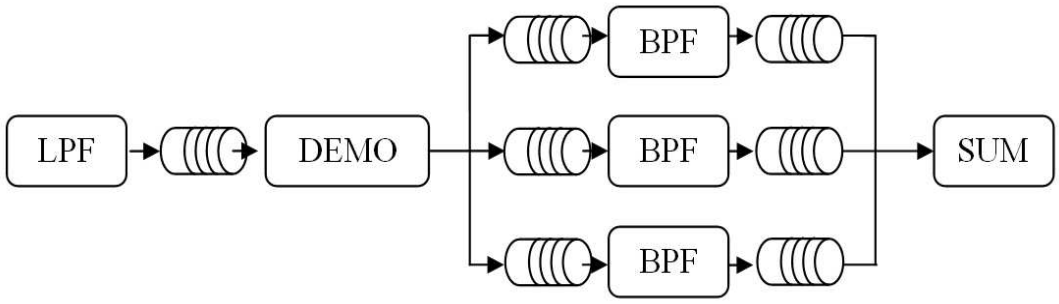


Figure 20: Schematic view of the SDR application

The first step in the processing phase is a low pass filter (LPF), and the resulting signal is demodulated (DEMOD) and shifted to the baseband. After that, the signal is forked in three branches to be equalized by three different band pass filters (BPF).

Finally a consumer (SUM) collects the data from every BPF. The communication between tasks is done using FIFO queues that transfer the data. Each task is allocated in a different processor during the load of the application.

Then, the policy implemented in the OS migrates the tasks depending on the temperatures of the cores. Figure 20 shows a schematic view of this application an the relations among the processing steps.

Evaluation of the Policies

The task migration policies implemented in the OS kernel were applied to the benchmark and the pre-defined metrics were collected to perform the evaluation.

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 4. Our experimental work starts at this point setting a thermal unbalance that motivates the activation of the migration policies.

Core (Frequency)	Load (%)	Temperature (K)
Core 0 (533 MHz)	44	340
Core 1 (533 MHz)	83	339.5
Core 2 (266 MHz)	29	328.5

Table 4: Initial state of the system

In the second phase, when the execution of the application effectively 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 and several conclusions can be extracted. Figure 21 shows the results of this analysis for the (a) adaptive) and (b) migration policies, where all the images have been taken at the same execution time.

As can be seen, both policies start similarly, decreasing rapidly the presence of hot spots. As time evolves, the adaptive policy obtains lower temperature values and a more homogeneous thermal distribution. In our benchmark, all the cells in the floorplan are within a range of temperature of 5 degrees when the adaptive policy is applied. Similar results were found when the adaptive policy was compared with the other task migration techniques.

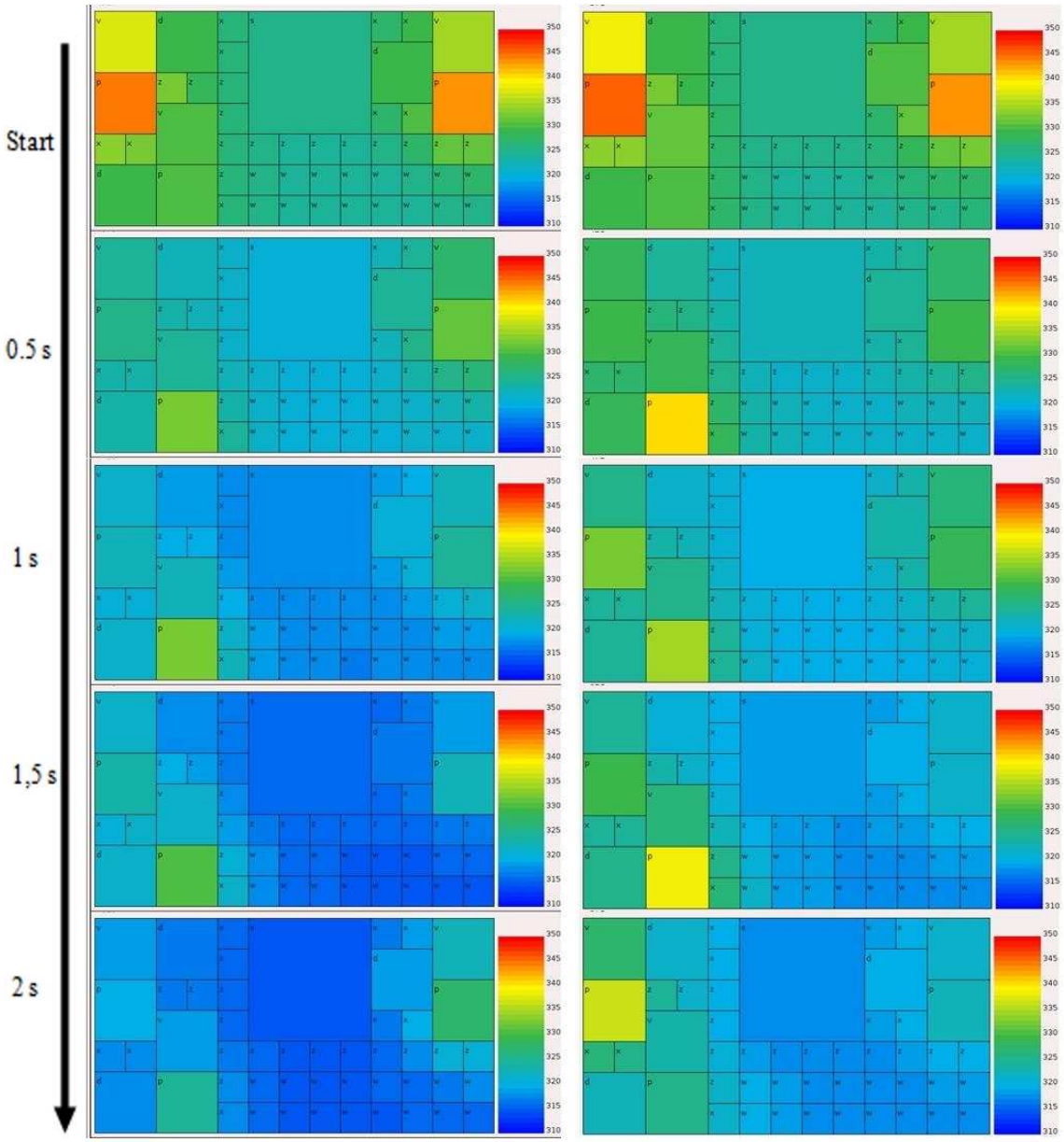


Figure 21: Thermal maps. (a) Adaptive; (b) migration

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 multiprocessor 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 22).

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 centred in the mean temperature (mean temperature 319.038 K with σ of only 2.53 K).

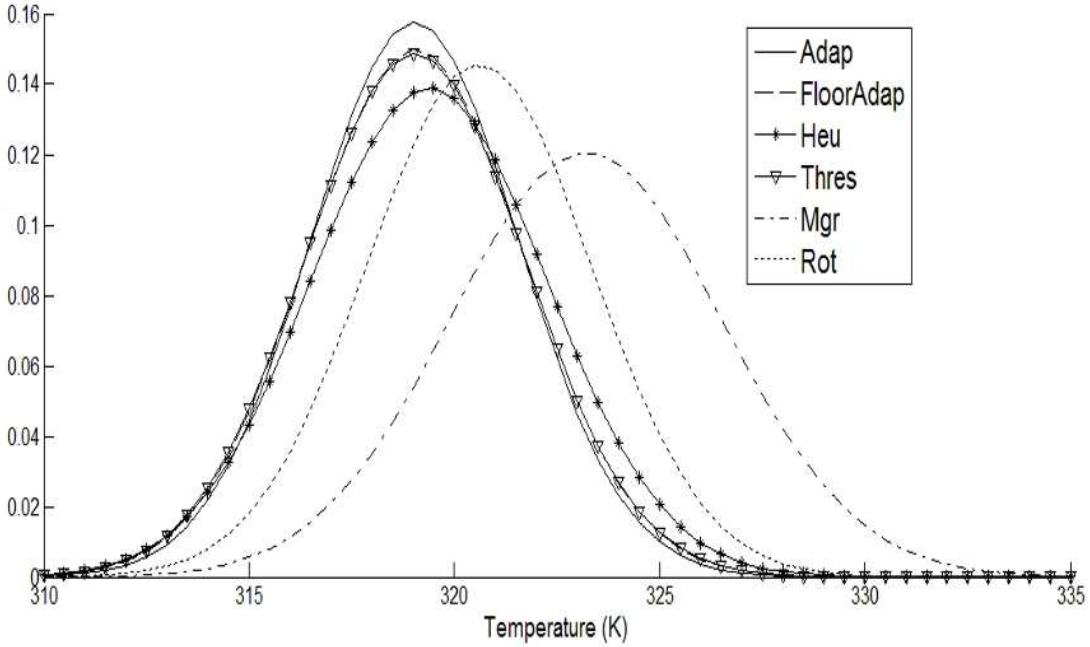


Figure 22: Normalized statistical distributions

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).

The other proposed techniques also obtain very positive results when the statistics are analyzed. For example, the implemented heuristic is able to decrease the mean temperature (319.38 K) and the sigma (2.87 K) of the statistical distribution when compared with rotation or migration.

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 23 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 [10] in terms of mean temperature and sigma of the thermal distribution, but they also decrease the impact on performance by 40% because of the less migrations per time unit are required.

	Adap.	FloorAdapt	Heu	Thres	Mgr	Rot
Overhead (%)	0.36	0.342	0.36	0.624	0.42	1.2

Table 5: Application performance overhead

Table 5 summarizes the performance overhead imposed by every task migration technique, where the minimum impact of our proposed policies can be observed. Besides this reduction could seem not enough, depending on the application could mean an important upgrade, not only because of performance gain, but also because of power consumption. It must be remembered that the more migrations take place the more power is consumed due to data transfer process.

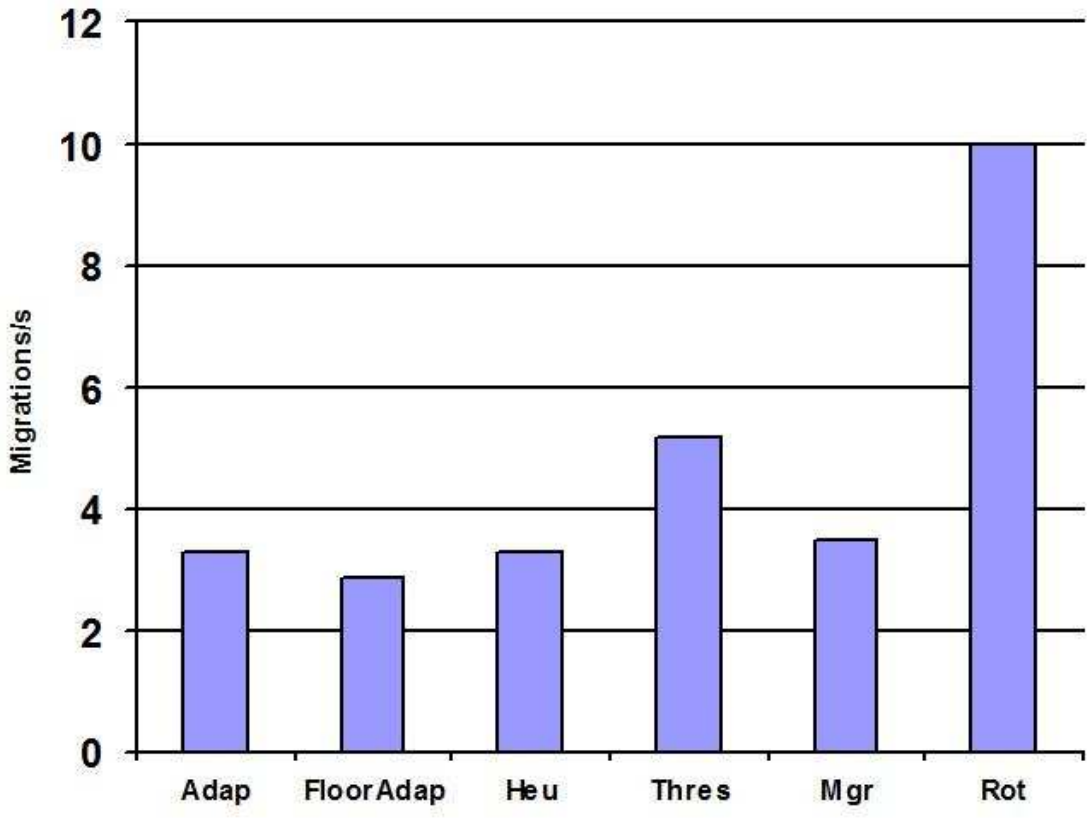


Figure 23: Number of migration per time unit

Finally, two factors with a very strong impact on the reliability of 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 328 K. Figure 24 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 has a strong impact on performance. The percentage of hot-spots is reduced to 1% and, therefore, the probability of system failure is minimized.

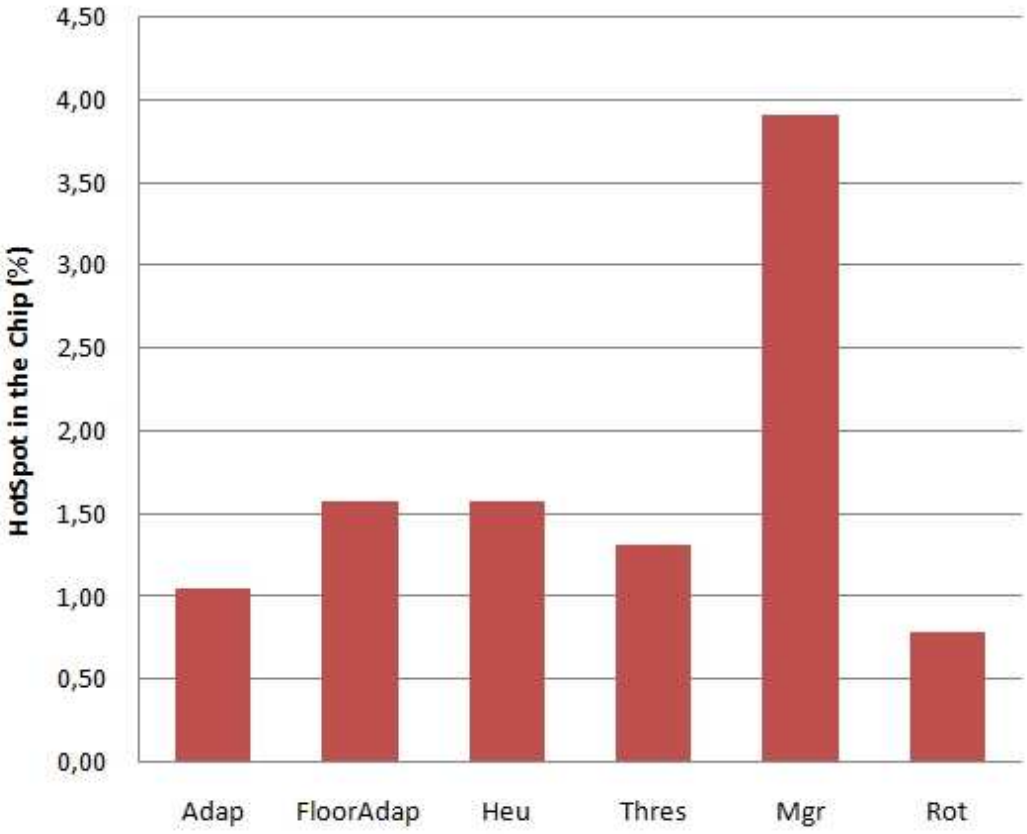


Figure 24:Percentage of hotspots in the chip

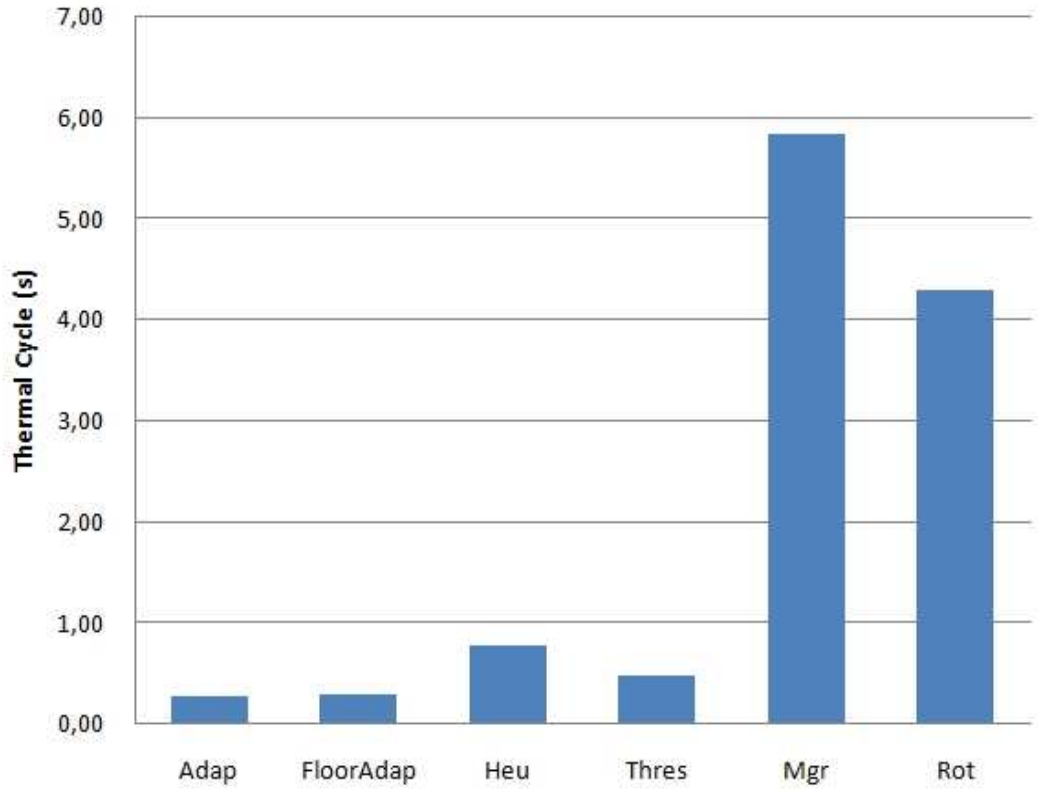


Figure 25: Thermal cyles

Figure 25 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 and with a reduced performance overhead.

All data presented we can reach the conclusion that floorplan aware policy is our best proposal because it reduces almost all the metrics to a minimum, compared with the other policies, with a low performance overhead.

Extrapolation to N-cores

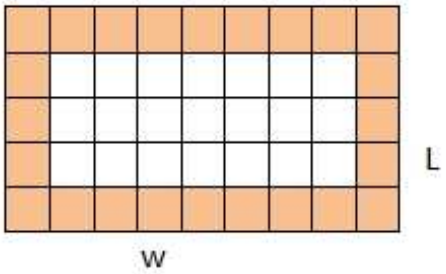
One of the main benefits of the task migration techniques that we have proposed in this work is their application to more complex systems with many processing units (N-cores).

While most of the previous techniques are not suitable for a system with a large number of cores because of the dramatic increase on the performance impact, our proposed techniques do no exhibit such behaviour and they are still suitable for

large systems. This subsection discusses this extrapolation to systems with N-cores.

If the size of the chip and the number of integrated cores are scaled accordingly, it can be predicted that the applicability of migration policies reaches a limit. This limit is reached due to the constrains imposed by the layout. As more cores are integrated in the floorplan, our policies achieve a mean value in the distribution curve of temperatures that depends on the floorplan of the chip and the number of cores.

Also, as the number of integrated cores is increased, the smaller the temperature deviation is. This fact is explained because the proposed task migration techniques have more target cores (cool cores) to balance the workload and, therefore, the temperature. In this way, they are able to make the cores working within a range of few degrees. Also, the complexity of the scheduler increases with the number of processors. However, as the selection of the task to migrate does not require the analysis of the task content (the migration is driven by those metrics exceeding a threshold), the scheduler does not constrain the scalability of the multiprocessor system.



$$N_{hifh} = \frac{l}{5}$$

$$N_{width} = \frac{l}{5}$$

$$w \leq \frac{13}{7} l$$

$$N = \frac{lw}{25}$$

$$N_{perim} = 2N_{width} + 2(N_{high} - 2)$$

$$N_{perim} \geq 0.5N$$

$$\frac{2l}{5} + 2\left(\frac{w}{5} - 2\right) \geq 0.5N$$

$$l \in [5,25]$$

$$N \leq 45$$

Figure 26: Floorplan extension

However, this trend also reaches a limit as the number of cores is increased. Supposing an extreme scenario where a 50% of the cores are hot spots, the task migration techniques are not able to eliminate these critic points. In this scenario,

there are too few processors that can be selected as a target for the tasks in the migration process. Therefore, the result of the migrations is a displacement of the hot spots in the chip area instead of their elimination. To help on the cooling process, the cores can be placed at the border of the chip to allow the thermal dissipation to the environment and be exploited by the floorplan-aware policy.

As the effective area of a single processor with its private memory is 5×5 cells in our set-up, an there is a relation of 13/7 between the length and the height of the chip, we can calculate the maximum number of processors that can be placed at the border of the chip when both the number of processors and the chip area are scaled linearly. Figure 26 shows the described set-up and clarifies the calculus. N_{high} is the number of processors in the Y-axis, N_{width} is the number of processors in the X-axis, N is the number of processors in the chip and N_{perim} is the number of processors placed in the border.

As can be extracted from the previous analysis, our task migration policies reach a limit in the optimization of the thermal profile when the number of cores exceeds 46.

Traditional approaches would be limited in their application at a smaller number of cores.

Chapter IV
Conclusions and Future Work

In this work, we have investigated and proposed 3 OS-level task migration policies for thermal management in embedded multi-processor 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 behaviour of the chip with a negligible performance overhead.

Our three proposals achieved great results in terms of temperature reduction and also in performance overhead introduction. All of them achieved these great results but floorplaning aware policy was our best result because it reduced the overhead introduced by task migration effect.

This work could be extended to new ways of design such as 3D floorplan placement. The thermal model that is used in 2D can be easily extended to a new dimension just adding a new thermal resistance that spreads heat in the other direction. Solving the equation system we would get temperatures for each layer of the chip.

Then a floorplan aware policy for task migration as the one we proposed here can be used to reduce power dissipation issues that these 3D structures present.

References

We will present:

- Bibliographic sources
- Figures index
- Tables index

Bibliography

[1]. Viredaz, M. A. andWallach, D. A. (2003) Power evaluation of a handheld computer. *IEEE Micro*, **23**, 66–74.

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

[3]. Semenov, O., Vassighi, A., and Sachdev, M. (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.

[4]. Borkar, S. (1999) Design challenges of technology scaling. *IEEE Micro*, **19**, 23–29.

[5]. Lu, Z.,Huang,W., Stan,M.R., Skadron,K., andLach, J. (2007) Interconnect lifetime prediction for reliability-aware systems. *IEEE Transactons on Very Large Scale Integration Systems*, **15**, 159–172.

[6]. Chaparro, P., Gonzalez, J., Magklis, G., Qiong, C., and Gonzalez, A. (2007) Understanding the thermal implications of multi-core architectures. *IEEE Transactions on Parallel and Distributed Systems*, **18**, 1055–1065.

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

[8]. 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 Great Lakes symposium on VLSI*, pp. 311–316.

[9]. 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.

[10].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 of the conference on Design, automation and test in Europe*, pp. 734–739.

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

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

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

[14].Nollet, V., Avasare, P., Mignolet, J.-Y., and Verkest, D. (2005) Low cost task migration initiation in a heterogeneous MP-SoC. *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 252–253.

[15].Bertozzi, S., Acquaviva, A., Bertozzi, D., and Poggiali, A.(2006) Supporting task migration in multi-processor systems on-chip: a feasibility study. *Proceedings of the conference on Design, automation and test in Europe*, pp. 15–20.

[16].Götz, M., Dittmann, F., and Xie, T. (2009) Dynamic relocation of hybrid tasks: Strategies and methodologies. *Microprocessors and Microsystems*, **33**, 81–90.

[17].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.

[18].Brião, E. W., Barcelos, D., Wronski, F., and Wagner, F. R. (2007) Impact of task migration in NoC-based MPSoCs for soft real-time applications. *Proceedings of the International Conference on Very Large Scale Integration*, pp. 296–299.

[19].Puschini, D., Clermidy, F., Benoit, P., Sassatelli, G., and Torres, L. (2008) Temperature-aware distributed run-time optimization on MP-SoC using game theory. *IEEE Computer Society Annual Symposium on VLSI.*

[20].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. *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, pp. 260–270.

[21].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.

[22].Yeo, I. and Kim, E. J. (2009) Temperature-aware scheduler based on thermal behaviour grouping in multicore systems. *Proceedings of the Conference on Design, Automation and Test in Europe.*

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

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

[25].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 Design Automation Conference*, pp. 746–749.

[26].Uclinux: Embedded linux/microcontroles project, 2006. **http://www.uclinux.org**

Figures Index

Tables Index

Equation Index