Performance/Reliability-Aware Resource Management for Many-Cores in Dark Silicon Era

Mohammad-Hashem Haghbayan, *Student Member, IEEE*, Antonio Miele, *Member, IEEE*, Amir M. Rahmani, *Member, IEEE*, Pasi Liljeberg, *Member, IEEE*, and Hannu Tenhunen, *Member, IEEE*

Abstract—Aggressive technology scaling has enabled the fabrication of many-core architectures while triggering challenges such as limited power budget and increased reliability issues, like aging phenomena. Dynamic power management and runtime mapping strategies can be utilized in such systems to achieve optimal performance while satisfying power constraints. However, lifetime reliability is generally neglected. We propose a novel lifetime reliability/performance-aware resource co-management approach for many-core architectures in the dark silicon era. The approach is based on a two-layered architecture, composed of a long-term runtime reliability controller and a short-term runtime mapping and resource management unit. The former evaluates the cores' aging status w.r.t. a target reference specified by the designer, and performs recovery actions on highly stressed cores by means of power capping. The aging status is utilized in runtime application mapping to maximize system performance while fulfilling reliability requirements and honoring the power budget. Experimental evaluation demonstrates the effectiveness of the proposed strategy, which outperforms most recent state-of-the-art contributions.

Index Terms—Dark silicon, lifetime reliability, many-core architectures, mapping, runtime resource management

1 Introduction

In the last decades, the aggressive technology scaling have brought to the massive miniaturization of transistors and the consequent integration of hundreds of cores within the same chip, leading to the definition of many-core architectures. However, with the end of Dennard scaling, supply voltage has not followed the same exponential scaling trend experienced with transistors [1]. Therefore, physical limits imposed by device packaging and cooling technology on peak power consumption and peak power density have made it impossible to power-on the entire chip at the same time, leading to the so-called *dark silicon* problem [1]. In practice, the designer has to specify a conservative Thermal Design Power (TDP) to avoid excessive temperatures potentially damaging transistor junctures; recently, Thermal Safe Power (TSP, [1]) has been also proposed to dynamically tune at runtime the available power budget according to the working configuration. The final effect of the dark silicon problem is the possibility to activate only a subset of the processing cores at the nominal voltage/frequency (VF) level, i.e., at maximum performance level, while the rest of the resources must remain power-gated, in a *dark* mode. According to projections of the International Technology Roadmap for Semiconductors (ITRS) drawn in 2013 [2], the percentage of dark silicon for a chip designed at 22 nm is around 50 percent while at 8 nm it will increase to 70 percent; this will represent a critical issue for near future many-core systems.

Power budgeting strategies (TDP or TSP) are able to avoid chip failures due to extreme power densities. However, they cannot handle reliability threatens in the long term period. In fact, modern devices use to experience higher temperature profiles, even if the peak values are within the tolerated guard bands. As stated in the ITRS report in 2011 [2], this trend together with the extreme downscaling of CMOS technology, has lead to an acceleration of aging and wear-out process of the chips. Eventually, aging mechanisms, such as time dependent dielectric breakdown (TDDB), negative bias temperature instability (NBTI), and electromigration (EM), lead to delay errors and, eventually, device breakdowns [3]. Past studies [4] have shown that failure mechanisms are exponentially dependent on temperature, and a $10\text{-}15^{\circ}C$ difference in operating temperature may result in a $2\times$ difference in the overall lifespan of a device.

Runtime resource management approaches are generally adopted to control the activities of many-core architectures. The main reasons are related to the high dynamicity of the workload, having applications entering and leaving the system with a unknown fashion. Moreover, applications are generally composed of a variable number of interconnected tasks presenting different computational characteristics and power profiles, and may expose some performance requirement. As a consequence, the runtime resource management

- *M.-H. Haghbayan, P. Liljeberg, and H. Tenhunen are with the Embedded Computer and Electronic Systems Laboratory, Department of Information Technology, University of Turku, Turku 20520, Finland. E-mail: {mohhag, pakrli, hatenhu}@utu.fi.*
- *A. Miele is with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano 20133, Italy. E-mail: antonio.miele@polimi.it.*
- *A. M. Rahmani is with the Department of Computer Science, University of California, Irvine 92697, and Institute of Computer Technology, TU Wien, Vienna 1040, Austria. E-mail: amirr1@uci.edu.*

Manuscript received 5 Sept. 2016; revised 20 Feb. 2017; accepted 13 Mar. 2017. Date of publication 4 Apr. 2017; date of current version 15 Aug. 2017. Recommended for acceptance by D. Kagaris.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2017.2691009

is in charge of achieving the optimal trade-off between the workload performance and the system power consumption. In this perspective, various dark-silicon-aware strategies for application mapping and power management have been proposed in the past (e.g., [1], [5], [6], [7]), but very few ones (e.g., [8], [9], [10], [11]) consider lifetime reliability issues. Indeed, stress-agnostic workload distribution may lead to an unbalanced aging among the cores thus causing a reduction of the lifetime of the overall system. On the other side, dark silicon may represent a new opportunity: in fact, the abundance of cores and the infeasibility to use all of them at the same time provide a unique opportunity for the runtime management to spread the utilization stress among the cores to prolong the system lifetime. In conclusion, we claim that *a paradigm shift from the conventional performance-centric runtime resource management to performance-reliability co-management is inevitable in many-core systems designed for the dark silicon era*.

Given these motivations, this paper proposes a novel reliability-aware runtime resource management approach for many-core systems to deal with the trade-off between workload performance and architecture aging while honoring the given power budget. The approach is an enhancement of the state-of-the-art runtime resource management layer i) by introducing a novel runtime reliability analysis unit estimating the aging status of each core in the architecture and computing a set of metrics showing the related reliability trend over the time, and ii) by extending the nominal application mapping and resource management scheme to consider also the reliability metrics and perform a balancing between workload performance and system's reliability in order to meet the required lifetime target. A preliminary proposal of this approach has been presented in [10] by considering only the mapping step. The key contributions of the more mature version of the framework that we propose here are the following:

- Proposing a more mature two-step application mapping approach which considers reliability metrics w.r.t. a lifetime target and the current VF map of the architecture to balance the performance/reliability trade-off while fulfilling the power budget.
- Defining a maximum VF capping strategy compliant with state-of-the-art reliability-agnostic power management approaches to unstress specific areas of the device that have been aged faster than the prevision.
- Presenting a more advanced reliability analysis unit with a detailed discussion of the reliability monitor.
- Presenting an extensive experimental evaluation revealing that the proposed approach can carefully guarantee the required lifetime of the chip for different power management strategies in long-term with a negligible performance penalty.

The paper is organized as follows: Section 2 briefly discusses the preliminaries on the many-core system and the adopted reliability model. Then, the subsequent Section 3 motivates this work by showing the potentialities of the exploitation of dark silicon for dynamic reliability management. Related work about dynamic reliability management in many-cores is covered in Section 4. The proposed approach is discussed in details in Section 5. Experimental results are provided and discussed in Section 6 and, finally, Section 7 concludes the paper.

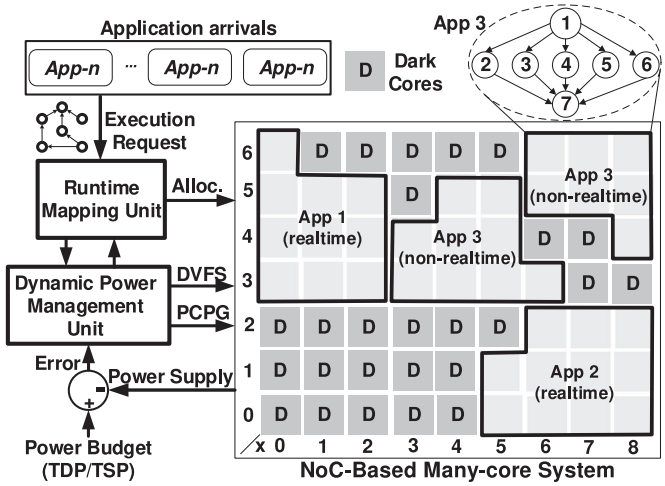


Fig. 1. The target platform: architecture, RRM layer and running workload.

2 Background

We briefly introduce here the required background on the many-core system and the reliability model.

2.1 The Many-Core Architecture

The target platform is the modern many-core architecture, such as the Intel SCC [12], the Kalray MPPA many-core [13], or the Adapteva Epiphany [14]. All these platforms present a similar Non-Uniform Memory Access (NUMA) architecture, shown in Fig. 1, consisting of a 2D mesh of homogeneous processing nodes interconnected via a Network-on-Chip (NoC) infrastructure. In the specific model we consider, as in [1], [5], [6], [7], each node (or core) contains a single processor provided with private instruction and data memories and a NoC network interface. Finally, the platform is connected to a host machine, controlling all its activities; for instance, in Intel SCC the Management Console Personal Computer (MCPC) manages the 48-core system via PCI-Express.

Many-core architectures are generally employed in High Performance Computing (HPC) and embedded HPC scenarios to accelerate computational-intensive applications such as image, video, or streaming processing. Some interesting use cases discussed in [13] are the autonomous driving or the cryptography acceleration. The commonly-adopted programming model is the dataflow one (as reported in [12] and [14] for Intel SCC and Adapteva Epiphany, respectively) that represents the application through a direct acyclic task graph [5], [6], [7], as shown in the top-right part of Fig. 1. In this model, the task is a single function/portion of code requiring specific input data, provided by precedent tasks, and producing specific output data, transmitted to the subsequent tasks, as described by the edges in the graph. To be executed, an application has to be dispatched on the grid of processing nodes; each task is *mapped* on a single idle node, i.e., not executing any other task. Hence, no multi-tasking is assumed at node level; in fact, as stated by Intel in 2011 [12], given the abundance of cores, a one-to-one mapping may ease the execution management. For similar reasons, task migration is also not supported. This solution has been later confirmed for the subsequent platforms available on the

market. Then, the execution model states that a task is run in a non-preemptive way as soon as all predecessor tasks have been completed and input data received. Communication is performed by means of messages passing based on the specific protocol adopted by the NoC infrastructure.

A specific *Runtime Resource Management layer* (RRM layer, [1]) is loaded on top of the discussed architecture to handle two relevant issues in the system's activities: i) variable workload, and ii) the limited power budget. Many-core architectures work in high evolving working scenarios with applications entering and leaving the system with an unknown trend; nevertheless, applications are highly heterogeneous in terms of size and shape of the task graph and may expose Quality of Service (QoS) requirements, expressed in terms of minimum throughput or latency to be satisfied. For this reason, a *Runtime Mapping unit* (RTM unit), a control routine running on the host machine, receives the execution requests of the various users and decides at runtime which group of resources to reserve for the each issued application depending on the available units and power budget. In case of unavailability of the minimum amount of processing resources, the request is stored in a ready list. To dominate the complexity of this phase, the RTM unit usually acts in two steps: i) *region selection*, that finds a set of neighboring idle cores to be reserved for the new application, and ii) *task mapping*, that dispatches the tasks of a single application onto the selected region.

Due to the dark silicon phenomenon, not all the available cores in the architecture can be switched on at the same time. To handle this issue, a second control routine, called *Dynamic Power Management unit* (DPM unit), is executed on the host machine. The unit implements a feedback control loop that receives the available power budget in terms of TDP or TSP, and analyzes the current power consumption for the architecture (related to the active cores and running applications). The DPM unit actuates on available per-core power-gating (PCPG) and dynamic voltage and frequency scaling (DVFS) knobs to modulate the power utilization at node granularity. At the same time, the unit informs the RTM one on the possibility to accommodate incoming applications, based on an estimation of their power necessities. Finally, when using DVFS actuation, the DPM unit transmits also the current VF configuration. This piece of information is relevant to map each application on a group of nodes tuned with a minimum VF level able to guarantee the expressed QoS requirements [15].

2.2 Reliability Model

The design of aging sensors is a widely explored research line in the last decade (e.g., [16]). However, they are not currently available in commercial devices. For this reason, as in various previous work (e.g., [17], [18], [19]), we here adopt the classical stochastic model for lifetime reliability generally employed in systems engineering and also for electronic devices, as stated in [3] by JEDEC Solid State Technology Association. The lifetime reliability of a system, $R(t)$, is expressed as the probability that the system has been operational until t. $R(t)$ formula for a single digital component, such as a processing core, is modeled by means of the Weibull distribution

$$R(t) = e^{-\left(\frac{t}{\alpha(T)}\right)^{\beta}} \tag{1}$$

being t the current instant of time (generally measured in hours), T the constant worst-case processor temperature (Kelvin degrees), β the Weibull slope parameter, and $\alpha(T)$ the scale parameter, or aging rate. The $\alpha(T)$ parameter formulation depends on the considered wear-out mechanisms (e.g., EM, TDDB or NBTI). As an example, in this paper we consider the EM model, where $\alpha(T)$ is modeled according to the Black's equation

$$\alpha(T)_{EM} = \frac{A_0(J - J_{crit})^{-n} e^{\frac{E_a}{kT}}}{\Gamma\left(1 + \frac{1}{\beta}\right)} \tag{2}$$

where A_0 is a process-dependent constant, J the current density, J_{crit} the critical current density for EM, E_a the EM activation energy (a constant value), k the Boltzmann's constant, n a material-dependent constant, and Γ the gamma function.

For the sake of simplicity, Equation (1) considers only a constant temperature. This aspect may cause pessimistic non-accurate evaluation of the reliability especially when the focus is on the analysis of a system with a variable workload (which causes considerable temperature variations). Therefore, as shown in [20], Equation (1) can be enhanced to consider temperature variations

$$R(t) = e^{-\left(\sum_{j=1}^{i} \frac{\tau_j}{\alpha_j(T)}\right)^{\beta}} \tag{3}$$

where τ_j represents the duration of each period of time with constant steady-state temperature T_j in the core up to time t (i.e., $t = \sum_{j=1}^{i} \tau_i$).

When the system integrates various units, the overall lifetime reliability $R_s(t)$ is obtained by combining the $R_i(t)$ of the single parts as series, parallel and K-out-of-N systems based on the architecture topology [20]. However, in this work we will not adopt such complex formulation since we aim at analyzing the system at node granularity level.

Finally, given this lifetime reliability model, the average lifetime of the system is estimated in terms of its Mean Time To Failure (MTTF)

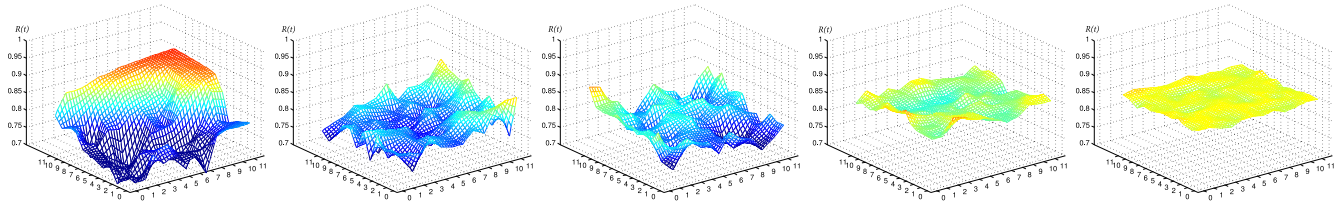
$$MTTF = \int_0^{\infty} R(t)dt. \tag{4}$$

3 Motivations

Motivations at the basis of this work are discussed here. We present a systematic and in-depth analysis on the factors affecting the system's aging. Then, we define how reliability requirements are defined in this work.

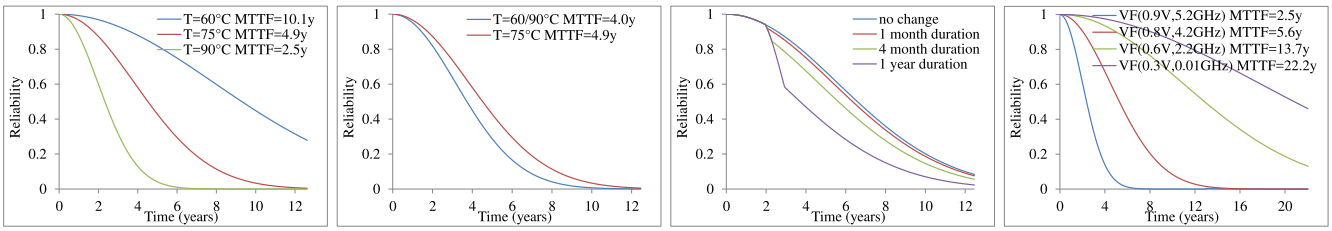
3.1 Effects of RTM and DPM Decisions on the Aging

We analyzed the most relevant dynamic mapping strategies for many-core systems proposed in the past, namely NN [5], SHiC [6], and MapPro [7], in terms of the aging effects on the various cores. A 12×12 many-core system was considered, and EM mechanism was characterized as in [20]. As shown in Figs. 2a, 2b and 2c, after a certain amount of time (i.e., 4 years) each strategy causes a different aging (computed with Equation (3)) on the various



(a) Using NN [5] (b) Using SHiC [6] (c) Using MapPro [7] (d) Using reliability-aware mapping (e) Using reliability-aware mapping and VF tuning

Fig. 2. The effect of different runtime mapping approached on cores reliability after 4 years.



(a) Core reliability in function of the temperature (b) Core reliability in a varying scenario (c) Effects of a drastic change of the temperature at time 2 year (d) VF effect on core aging

Fig. 3. Accurate analysis of the causes of a core aging.

processing units according to how they were used by the mapping process. In all these situations, the aging is highly unbalanced since such strategies are reliability-agnostic. MTTF of the most aged core in each scenario is equal to 5.4, 6.0, 6.0 years, respectively.

Indeed, if we carefully consider how such reliability-agnostic strategies work, we may notice that when there are various regions of cores suitable for executing the newly-arrived applications, they take decisions only by means of metrics prioritizing the minimization of the network congestion and mapping fragmentation to achieve a higher performance. Potentially there may be several suitable regions, but the selection will be performed in a reliability-agnostic way. As a consequence, due to single objective management strategy in these algorithms, some cores will be more frequently selected and will age faster. At the opposite, a smarter algorithm considering performance and aging together may achieve the same level of performance and at the same time balance the stress among the cores. Fig. 2d shows the results that we may achieve by considering also a reliability metric within MapPro. In this case, the MTTF can be improved by up to 9.2 years. Nonetheless, such result can be further improved when considering other actuation knobs such as the tuning of maximum VF levels (Fig. 2e with a MTTF of 10.7 years).

To better understand the relationship among utilization, temperature and lifetime reliability in order to properly define a reliability-aware RRM approach, let us consider some simpler scenarios. Fig. 3a shows the reliability of a single core acting a constant steady-state temperature. As already stated, this plot shows how MTTF as an exponential relationship with the temperature. As an obvious conclusion, the lower the temperature, the longer the lifetime.

In the second scenario, depicted in Fig. 3b, we analyze the reliability of a core working in two alternative states causing a periodic temperature change between $60°C$ and $90°C$, and we compare it with the one of the same core working at the steady-state temperature of $75°C$, i.e., the average of the previous two values. As we can see the second curve represents a considerable overestimation of the former one. As a conclusion, as also commented in [19], [21], aging cannot be accurately controlled by monitoring the average temperature but it is necessary to act directly on the reliability metric.

In the third scenario, we consider a core working at a constant temperature of $60°C$ and having a drastic change in the workload at the time when the core's age equals to 2 years which causes an increase of the temperature to $90°C$. As shown in Fig. 3c, this change has a visible effect on the overall reliability only when its duration is considerably long (in terms of weeks or months) otherwise it is negligible. Therefore, single fluctuations are in general imperceptible, while the long term temperature profile will have a perceptible effect on the aging trend. As a consequence, short term performance bursts within the power bounds have no dramatic effects on the system reliability.

Finally, when acting on the operating VF level, the core is subject to considerable temperature variations. In Fig. 3d, the plot shows the various reliability curves related to the different VF levels of a core. If we analyze the MTTFs, we may notice that while performance scale linearly, there is an exponential falling trend in the lifetime. Therefore, the VF knob also offers the possibility to control the aging process, however it may degrade the performance as a drawback.

Putting together all these facts, *we believe that resource management for modern and dark silicon aware many-core systems necessitates an efficient multi-objective feedback-based approach which considers per-core power, performance, thermal and lifetime measurements all together. In other words, it needs to be able to couple advanced power management knobs such as DVFS and PCPG with dynamic application mapping techniques to mutually co-manage performance and reliability.*

3.2 Reliability Target

Another consideration is related to how we may define a reliability goal pursued by the proposed reliability-driver

RRM approach. In many past works [22], [23], the reliability goal was defined as the optimization of the MTTF. However, this formulation suffers from various issues w.r.t. the considered working scenario.

First of all, MTTF computation requires the $R(t)$ curve to be known in each instant of time, from 0 to ∞ as stated in Equation (4). This precondition holds only when the system presents a predictable aging trend, for instance, when the system has a periodic or fixed activity plan defined at design-time (e.g., [24]). On the other hand, when considering highly dynamic scenarios, it is not possible to foresee the actual values of the $R(t)$ model in the future. Some past runtime reliability management approaches (such as [22], [23]) compute MTTF according to a very simple prediction of the future aging trend based on the previous history and by using approximated formulas. Indeed such computations lead to completely unreliable MTTF values.

As discussed by ReliaSoft (a global leader in reliability engineering software) in [25], another limitation of the MTTF is its incapability to capture the shape of the $R(t)$ model over time. Indeed, curves presenting the same MTTF may have different distribution over time. If we consider the fact that in most of the situations the duration of the service period of system is almost known and established at the deployment step by the system architect or the owner company, the maximization of the MTTF beyond such a period does not provide any additional advantage. At the opposite, as stated in [25], it would be useful to maximize the value of the reliability model within the service period, or at least to fulfill a minimum threshold, to improve the probabilities of the system to not fail before the system is retired. For this reason, other past works [18] adopt an alternative approach for defining the reliability target by setting a given reliability level $R(t_{target})$ the system must have at the end of the envisioned lifetime t_{target}. For instance, the reliability target can be specified as follow: *At the end of the working life, estimated in* $t_{target} = 10$ *years, the system must have at least a reliability of* $R(t_{target}) = 45\%$.

In conclusion, we select the target lifetime $R(t_{target})$ as the main metric in our proposed reliability management technique. We show that using co-management of resource and power, our approach can provide specified target lifetime in long term while satisfying other constraints such as power budget in short term with a negligible performance penalty.

4 Related Work

The highly evolving and unpredictable characteristics of the workload in the on-demand working scenario has led to the definition of *Dynamic Reliability Management* (DRM) approaches for multi-/many-core systems. The DRM idea has been defined for the first time in [22], where DVFS has been employed for mitigating the aging of a single processor. However, to simplify the computations, the approach considers a reliability model based on an exponential failure distributions, which is not realistic. Moreover, the DRM strategy is quite immature, and considers single-processor systems. Later, further similar approaches (e.g., [23]) acting on DVFS and considering single-processor systems have been proposed; however they also suffer from similar limitations.

Many DRM approaches have been proposed for bus-based multi-core systems. Some examples are [26], [27], which act on job scheduling to reduce aging effects, and [17], [18], which exploit also DVFS tuning. However, many-core systems have a very different architecture, programming paradigm and execution model from the shared-memory bus-based multi-core counterpart. In particular, since they are based on a NUMA message-passing architecture, application mapping plays a more crucial role in many-core systems and presents specific characteristics for that architecture. Therefore, as also empirically shown in the experimental campaigns discussed in Section 6, a straightforward porting of previous multi-core-based DRM policies to the considered scenario would lead to ineffective solutions due to the lack of tight connection with the application mapping.

The impact of lifetime reliability on runtime resource management in NoC-based architectures has been considered in some works (e.g., [19], [21], [28], [29], [30], [31]). The strategy in [28] acts on DVFS to control the aging of the units in a many-core architecture. However, single-threaded applications are considered, and the application mapping and power budgeting are not completely addressed. The work presented in [21] defines a reliability-aware mapping approach, however, it also suffers from similar limitations of the previous ones. Then, the work in [29] proposes a systematic analysis of various mapping policies w.r.t. aging issues; examples of considered policies are the uniform aging balancing, the adoption of spare units or the rotation of a single spare region. Once again, single-thread applications are considered and power management is not addressed. Indeed such DRM strategies cannot be easily integrated into the complex RRM layer of many-core systems. As we also demonstrate in the experimental evaluation in Section 6, the straightforward integration of existing approaches is not effective, since they only consider a part of the complex picture, and often, they have partially contradicting objectives with the RRM policies.

The approaches proposed in [30], [31] define migration controllers that move tasks from elder cores to younger ones. However, the approach is too fine-grained as device reliability changes very slowly over time (in the order of days). Therefore, a periodic migration of the workload would be necessary only for applications lasting for days or weeks. Nevertheless the approaches which are based on task migration, would lead to non-optimal applications' performance. Another mapping approach has been presented in [19]; the mapping algorithm is quite limited since it performs an almost-exhaustive exploration of the solution space and, moreover, it does not consider performance optimization in the mapping, and an enhancement in this direction seems to be infeasible. A dynamic mapping approach based on a many-core partitioning is introduced in [32]; even though the idea is interesting, it presents some limitations as it does not take into account power constraints.

Finally, there exists very few contributions for addressing this issue [8], [9], [10], [11]. In [8], the mapping policy considers also the current aging and the process variation status of the cores; however, again a shared-memory architecture is considered and mapping policy does not consider the topology and related communication issues in threads distribution. The work in [9] proposes a machine-learning strategy to perform aging-aware mapping. A simplified exponential model is used for lifetime reliability. Nevertheless, also in

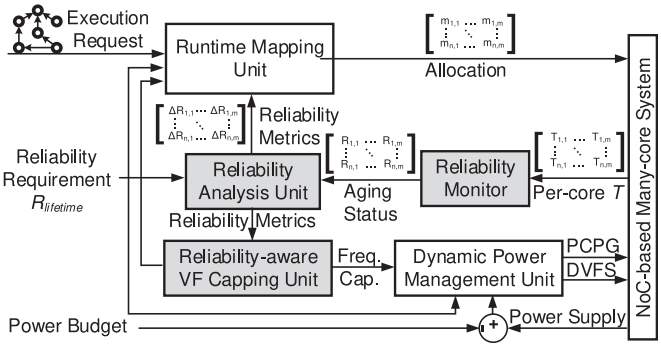


Fig. 4. The proposed reliability-aware RRM layer.

this case a shared-memory architecture is considered, and, as a consequence, the mapping strategy is quite simplistic. Nevertheless, none of these approaches take into account DVFS tuning for aging mitigation.

Our first attempt to define a dark silicon and reliability-aware mapping strategy for many-core architectures has been presented in [10]; as discussed in Section 1, we will here enhance that work. Orthogonally, in [11] we have also addressed reliability issues only in power management. However, in that approach the reliability is considered as a secondary metric to be optimized; instead, in this work, we perform a resource and power co-management to primarily guarantee lifetime requirements; therefore, the two approaches cannot be compared. Nonetheless, the approach in [11] is very specific and is "hardcoded" in an advanced DVFS strategy; therefore, it cannot be employed with different strategies for instance using only PCPG knob.

5 The Proposed Reliability-Aware Resource Management Approach

Fig. 4 shows the framework of the proposed approach. It is an enhancement of the RRM layer (introduced in Section 2.1) to handle aging issues concurrently to the nominal application mapping and power management.

The enhanced RRM layer is organized into two main parts being in charge of the workload execution and lifetime reliability management, respectively. Indeed, such partitioning of the activities is motivated by the fact that they follow two very different time horizons: application mapping and power management activities are performed with a short-term frequency, since applications can be issued every moment and they last for a period ranging from some seconds to few hours, while reliability can be managed with long-term decisions, since the aging of a system relatively is a slow phenomenon and has perceptible effects over epochs lasting for days, weeks or even months, as discussed in Section 3.1.

The central part of the framework in Fig. 4 (filled in gray color) represents the *long-term controller*, which performs the reliability management and contains the *Reliability Monitor*, the *Reliability Analysis Unit* and *Reliability-aware VF Capping Unit*. The former is an utility unit that computes that aging status of each node within the architecture by continuously reading the temperature from the per-core sensors and applying the adopted reliability model. Then, the Reliability Analysis Unit monitors the aging status of the various nodes according to the information gathered by the Reliability Monitor. In particular, according to the reliability requirement $R(t_{target})$ at the end of the lifetime t_{target} provided at the beginning of the service life by the system architect, it computes the target reliability curve. This curve represents an aging reference showing how fast each node should age in order to fulfill the given reliability requirement. Then, the unit periodically analyzes the current reliability value of each node w.r.t. the target aging reference to compute specific reliability metrics describing the aging trend to be used in the mapping decisions. Finally, the Reliability-aware VF Capping Unit takes additional recovery actions to unstress nodes that have already consumed the available "reliability budget", i.e., their reliability is considerably below the reference curve. Its main strategy is to cut maximum VF levels of selected nodes to reduce temperature peaks, and, consequently, slow down the aging trend.

The rest of Fig. 4 represents the *short-term controller*, containing the classical set of units devoted to the management of the nominal activities of the system. In this proposal, such units have been specifically enhanced to take also into account the reliability metrics provided by the long-term controller in the decision process. In particular, the reliability metrics are used as weight in the mapping decisions in order to prefer the utilization of younger nodes; while power management needs to take into account the reliability-driven maximum VF configuration. The internals of the various units are discussed in details in the following sections.

5.1 Reliability Monitor

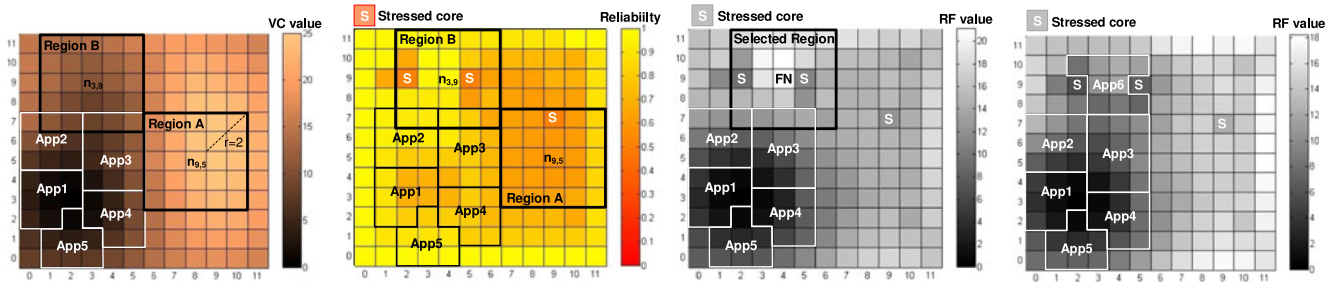
The *Reliability Monitor* estimates the lifetime reliability of each core within the architecture based on the model in Equation (3). Since we assume a single sensor to be integrated in each processing node, as it commonly happens in modern architectures such as Intel SCC, the reliability model is applied at per-core granularity level. In details, the node's temperature is sampled with a time step (few milliseconds/seconds) in which we may assume to be steady-state, and the instantaneous aging rate α_i is computed according to Equation (2). To keep track of the aging trend, it is necessary to save the overall profile of the α_i values represented in the expression at the exponent of Equation (3), that is $\sum_{j=1}^{i} \frac{\tau_j}{\alpha_j(T)}$. Therefore, we use a variable $A(t)$, initialized to 0 at the beginning of the operational life, and incremented at each time step with the current aging value as follows:

$$A(t_i) = A(t_{i-1}) + \frac{t_i - t_{i-1}}{\alpha_{t-1}(T)} \tag{5}$$

where $t_i - t_{i-1}$ is the interval (generally measured in hours) between two subsequent measurements. Thus, in each instant of time, $A(t)$ can be used to compute the reliability value $R(t)$ of the core.

5.2 Reliability Analysis Unit

The *Reliability Analysis Unit* monitors the cores' aging status and provides reliability metrics to the short-term controller. At the beginning of the operational life, the unit defines a target reliability curve $R_{target}(t)$, called *aging reference*, according to the required reliability target $R_{lifetime}$ at the given lifetime $t_{lifetime}$. This curve represents a sort of reliability budget each core is provided with, i.e., how ideally the reliability of a



(a) Region selection based on only network characteristics. (b) Reliability distribution of the cores (c) Region selection based on RF distribution (d) Mapping the application while considering too stressed cores

Fig. 5. An example to show reliability-aware mapping of a new coming application.

node working at a steady-state temperature should evolve during the operational life at least to fulfill the reliability requirement. $R_{target}(t)$ is defined for each node as

$$R_{target}(t) = e^{-\left(\frac{t}{\alpha_{target}}\right)^{\beta}} \tag{6}$$

where, α_{target} is obtained by inverting Equation (1) as follows:

$$\alpha_{target} = \frac{t_{lifetime}}{\left(-\ln(R_{lifetime})\right)^{1/\beta}}. \tag{7}$$

Then, the unit computes for each node $n_{w,h}$ the difference of the current reliability and its target value

$$\Delta R_{w,h}(t) = R_{target}(t) - R_{w,h}(t). \tag{8}$$

This metric shows how each core is using the reliability budget available according to the final reliability target:

- if $\Delta R_{w,h}(t) > 0$, the core has been aged much more than the expected reference value; therefore, it needs to be offloaded by avoiding its usage to restore from the excessive experienced stress.
- if $\Delta R_{w,h}(t) \leq 0$, the core is younger than the expected age, and, thus, it can experience a higher stress in the future.

Finally, all $\Delta R_{w,h}(t)$ values are transmitted to the RTM unit and the Reliability-aware VF Capping one.

5.3 Reliability-Aware RTM Unit

The proposed *RTM Unit* aims at optimizing the overall performance of the system (in terms of number of completed applications over time), while guaranteeing reliability requirements. In the design of the RTM unit, we followed the state-of-the-art two-step approach, consisting in region selection and task mapping. We enhanced the two most effective strategies proposed in the literature, called MapPro [7] and CoNA [33], respectively, to be provided with reliability awareness.

In the first step, MapPro starts the region selection process with locating a suitable *first node*. This node represents the center of an almost squared regions containing at least the minimum number of cores required for accommodating application tasks. This process is here enhanced to consider also the aging status of the various cores provided by the Reliability Analysis Unit.

The original MapPro strategy defines a square region in terms of a first node $n_{w,h}$ and a radius r. Moreover, a vicinity counter metric VC measures the suitability of each region according to its characteristics such as the size and the fragmentation. As an example, Fig. 5a shows two regions, A and B, located in different positions and having the same radius equal to 2. We here replace the original vicinity counter metric used to identify the best region with a novel metric called *Reliability Factor* (RF), incorporating reliability awareness. RF is defined on a square region identified by the first node $n_{w,h}$ and radius d as

$$RF^{r}_{w,h}(t) = \sum_{i=i-r}^{i+r} \sum_{j=j-r}^{j+r} Wn_{i,j} \cdot (r - d + 1) \cdot R_{i,j}(t) \tag{9}$$

where $R_{i,j}(t)$ is the reliability of core $n_{i,j}$ at time t, and d is the distance from an occupied core to the center ($r \geq d$). The first two terms, that are the original ones, aim at reducing mapping fragmentation and dispersion. In particular, $Wn_{i,j}$ is the weight of core $n_{i,j}$

$$Wn_{i,j} = \begin{cases} 1 & \text{if } n_{i,j} \text{ is unoccupied} \\ 0 & \text{if } n_{i,j} \text{ is occupied.} \end{cases} \tag{10}$$

It discourages the selection of regions where there are busy cores to avoid fragmentation. The second term pushes in the selection of the smaller square region, to leave space to the accommodation of other applications. Finally, the new $R_{i,j}(t)$ factor pushes to the selection of regions with younger cores. To this purpose, when a core has $\Delta R_{i,j}(t) > 0$, the actual $R_{i,j}(t)$ value is replaced with 0 in the formula, to penalize the usage of such core.

Indeed, RF represents a metric to express the number of free cores as well as determining the extent to which the region around the selected first node is reliable. Therefore, by using RF metric, the effect of occupied cores in the region on internal congestion and selecting a group of younger cores is considered together. For instance, a core that is occupied in the inner most square close to the first node has more impact on internal congestion than the ones that are occupied in outer squares, far from the first node. Therefore, the reliability of the node closer to center of the region (i.e., first node) affects the RF more than the ones that are farther away.

In Fig. 5, we exemplify the process of region selection by means of a scenario in which a new coming application has to be mapped on a system while five applications are in execution. Fig. 5a shows the traditional vicinity counter taking

into account only network characteristic; the filling color shows the suitability of each first node for the given radius $r = 2$ (the softer the color, the more suitable the first node). As it can be seen from the first node of Region A, comparing to the one of Region B, has a higher vicinity value which leads the mapping strategy to select $n_{9,5}$ as the first node. Fig. 5b depicts, with the filling color, the reliability distribution across the chip. It can be observed that the nodes are more aged in Region A in contrast to Region B. Moreover, nodes having $\Delta R > 0$ are tagged as stressed (S). The newly-defined SF metric combines VC and the reliability metric shown in Fig. 5b, thus resulting to the scenario in Fig. 5c where Region B gets higher chances to be selected as the candidate for task mapping.

Algorithm 1. Reliability-Aware Region Selection

Inputs: $newApp$: New application;
Outputs: Q: Selected mapping;
Variables: $appRadius$: Radius for $newApp$;
RF: Reliability factor;
$maxRF$: Regions with the maximum RF for each radius r;
$firstNode$: Currently selected first node;
Constants: $maxRadius$: Radius of the architecture grid;
Body:

```
1:  appRadius ← (√|newApp| − 1)/2;
2:  firstNode ← maxRF[appRadius];
3:  Q ← map(firstNode, newApp);
4:  //Updating RF value after mapping
5:  for each n_xy ∈ Q do
6:    for each core n_ij in the architecture do
7:      for r = 1 to maxRadius do
8:        r' ← maximum(|i − x|, |j − y|);
9:          if r − r' ≥ 0 then
10:           RF^r_ij ← RF^r_ij − (r − r') · R_xy(t);
11:           if RF^r_ij > maxRF[r] then
12:             maxRF[r] ← RF^r_ij;
```

Algorithm 1 shows the reliability-aware region selection strategy based on MapPro. The algorithm performs an immediate selection of the first node by using two specific data structures, RF and $maxRF$, storing the current status of the architecture. In particular, RF is a cubic matrix containing the reliability factors for all possible first nodes in each coordinate i, j in the architecture and by considering all possible radius r between 1 and $maxRadius$ (representing the radius of the architecture grid). It is worth mentioning that for some r value, the identified region may overcome the architecture bound. In such a case, hypothetical out-of-border cores are considered as busy to discourage selection of the region at the edge of the chip. Then, $maxRF$ is a lookup table containing for each radius r between 1 and $maxRadius$ the coordinates i, j of the first node currently having the highest square factor. RF and $maxRF$ are initialized at the beginning by considering an empty, new architecture.

By entrance of a new application, the algorithm computes the radius of the square region based on the size of the application, i.e., $appRadius$ (Line 1). Then, the maximum RF value from $maxRF$ lookup table is identified and transmitted to the task mapping function, $map()$ (Lines 2–3). When the tasks are mapped onto cores, RF and $maxRF$ data structures are proactively updated in order to allow the immediate mapping of the next entering application. To do this, we analyze the effect of each node selected by the $map()$ function for mapping the application on all the other nodes in the architecture. Therefore, the algorithm scans all positions in the RF matrix (Lines 5–7) and if the allocated node $n_{x,y}$ is within the region identified by the first node $n_{i,j}$ and radius r (Lines 8–9), its RF value will be updated by subtracting their current value from the weighted reliability of the occupied core according to Equation (9) (Line 10). Concurrently, the $maxRF$ for the current radius r is updated, if necessary (Lines 11–12).

A similar process to update RF and $maxRF$ is also executed for free nodes when an application leaves the system, with the difference of adding weighted reliability of such nodes instead of subtracting at Line 10.

Algorithm 2. Reliability-Aware Task Mapping

Inputs: App: Application;
$firstNode$: Selected First Node;
Outputs: Q: Selected mapping;
Variables: ΔR_{xy} : Reliability metric for n_{xy};
$Freq_{xy}$: Current frequency of n_{xy} received from DPM unit;
App_{Freq}: Minimum frequency for application QoS;
$mapFlag$: Mapping flag;
$Tasks$: Vector of tasks in App;
$Nodes$: Vector of architecture nodes;
$n_{recovery}$: Selected node for contingency mapping;
Body:

```
1:  App_Freq ← freq(App);
2:  Tasks ← CoNA_sort(App);
3:  for each t_i ∈ Tasks do
4:    Nodes ← not_busy_reliability_distance_sort(firstNode);
5:    mapFlag ← false;
6:    n_recovery ← None;
7:    while Nodes ≠ ∅ && !mapFlag do
8:      n_xy ← Nodes.pop_front();
9:      if ΔR_xy(t) ≤ 0 && App_Freq ≤ Freq_xy then
10:       Q[t_i] ← n_xy;
11:       mapFlag ← true;
12:     else if n_recovery ≠ None && App_Freq ≤ Freq_xy then
13:       n_recovery ← n_xy;
14:   if !mapFlag && n_recovery ≠ None then
15:     Q[t_i] ← n_recovery;
16:   else
17:     Q ← ∅;
18:     ABORT();
19: return Q;
```

After determining the first node, the $map()$ function (Line 2 in Algorithm 1) implements the second step of the RTM unit, i.e., the task mapping on the selected nodes. The designed strategy, shown in Algorithm 2, is an extension of CoNA, one of the most optimum state-of-the-art approach for inter-region mapping. In the first step, the algorithm gets the minimum frequency required by the current application to satisfy possibly specified QoS; as in [15], such value is precomputed according to a worst-case contiguous mapping and stored in a lookup table (Line 1). Then, application tasks are sorted in a vector according to the CoNA strategy. The overall idea of this strategy is that the optimal mapping is the one presenting lower communication congestion, since it is the dominant cause on the network

performance. Therefore, tasks sorting is based on the estimation of the amount of communication (from the most connected tasks to the least ones). In this way, the tasks having heavier communication signs are mapped near to the center (as discussed in the next), thus achieving lower average distance from the other tasks.

Then, for each task t_i, the algorithm looks for the best candidate node for mapping. Therefore, all free nodes are sorted and are scanned to select the first suitable one (Lines 4–13). The sorting key is the product of distance from the first node and reliability with the aim of preferring the selection of nodes being younger and closer to the center of region (i.e., first node). The selected node has to fulfill both reliability and performance requirements (Line 9). Otherwise, the algorithm looks also for a contingency solution; in particular, the first analyzed node satisfying only the frequency requirement can be candidate (Lines 12–14) and used for a contingency mapping only in case all the other nodes violate the reliability budget (Lines 14–15). In fact as noted in Fig. 3c, the usage of a stressed core for a small amount of time will not cause any visible additional worsening in its aging status. Finally, if no contingency mapping is identified for the current node, i.e., there is no node capable of guaranteeing the minimum VF level demanded by the application, the function is aborted and the application is re-inserted in the incoming queue for a new attempt in the near future (Lines 16–18).

To complete the discussion of the example, in Fig. 5d, the final mapping of the application is depicted. As it can be observed, stressed nodes have not been used for task execution. Moreover, after task mapping, the region selection strategy proactively updates the RF matrix (for the sake of simplicity the filling color represents again only the RF values for a radius $r = 2$).

5.4 Reliability-Aware Power Capping Unit

Dynamic power management generally performed in many-core systems is a complex process involving many parameters and figures of merit. One of the main considered aspects is to sustain performance while avoiding peak power violation. To do so, usually power management strategies tune different actuation knobs such as PCPG and DVFS to increase the system utilization which results in better concurrency and throughput. This results in a heterogeneous power distribution on the chip where some cores are power gated or operating at low or high frequency which results in unbalanced reliability of the cores. In such cases, while considering the system reliability, even though the DVFS reduces the power in some parts of the chip which results in reliability improvement on that part, increasing the utilization has negative effect on the efficiency of the resource management in terms of balancing the reliability.

For this reason, to add flexibility to the framework, we add a new external module to the DPM unit performing reliability-aware maximum VF capping. The new unit tunes in a long-term period the upper bound for core's operating VF based on the aging status. Such maximum VF levels are then transmitted to the nominal DPM unit. As demonstrated in the example in Fig. 3d, this method can limit the heating and the temperature peaks in too-aged nodes to perform a sort of stress recovery. At the same time, it leaves to the DPM unit the actual DVFS control to provide required performance. The only critical situations in which this unit affects system performance is caused by the attempt to map an application with a high QoS demand on a considerably-aged core. In fact, the mapping will fail since the core is not able to offer the minimum frequency level, and the application will be delayed as discussed in Section 5.3.

Algorithm 3. Reliability-Aware VF Capping

Inputs: ΔR: Reliability metrics;
Outputs: $maxVF_{ij}$: Maximum VF levels for the various cores;
Constants: Γ: Number of supported VF levels;
ΔR_{TH}: Threshold value for ΔR violation;
Body:
1: **for** each node n_{ij} in the architecture **do**
2: **if** $\Delta R_{ij}(t) > 0$ **then**
3: $maxVF_{ij} \leftarrow \Gamma - \left\lfloor \Gamma \cdot \frac{\Delta R_{ij}(t)}{\Delta R_{TH}} \right\rfloor$;
4: **if** $maxVF_{ij} < 1$ **then**
5: $VF_{ij} \leftarrow 1$;

Algorithm 3 describes the proposed reliability-aware VF capping process. The metric used to tune the VF level of each node is once again ΔR. The larger its value, the lower the set of maximum possible VF levels. For each core in the system not satisfying the reliability budget ($\Delta R_{ij} > 0$), the maximum VF level, $maxVF_{ij}$, is proportionally scaled down to the criticality of the violation (Line 3). In particular, $maxVF_{ij}$ contains the index of the available VF levels, numbered from minimum 1 to the maximum Γ (we assume VF levels to be uniformly distributed in the frequency range and increasingly sorted). The second constant is ΔR_{th} which represents the reference to normalize the criticality of reliability budget violation. In case of excessive violations, $maxVF_{ij}$ may be assigned with a negative number. In that case, its value is restored to the minimum possible, i.e., 1. Then, $maxVF_{ij}$ is transmitted to nominal DPM unit. Finally, for the sake of simplicity in the RRM layer, VF capping is performed only when the core is idle.

Two final comments can be drawn on the presented approach. First, the proposed approach is able to have a highly beneficial effects for all aging mechanisms described with the model presented in Section 2.2, because their $\alpha(T)$ formula is mainly an exponential function of the temperature value (e.g., Equation (2) for EM). As discussed also in [21], the only exception is Thermal Cycling; in fact, it depends not only on the temperature value but also on the amplitude and the frequency of the temperature fluctuations. In this case, the benefits are contained to the fact that limiting the usage of stressed cores would limit also the amplitude and the frequency of thermal cycles. A more specific approach targeting the minimization of cycles would be an interesting extension of this work.

Finally, the proposed approach does not impose any performance overhead to the executed workload due to the fact that all the four units of the RRM layer are executed on the host machine, therefore separately operating from the workload execution on the many-core architecture. Moreover, an analysis of their computational complexity is relevant to understand the performance impact of the four

TABLE 1
Summary of Acronyms in Section 6

DVFS	Dynamic voltage frequency scaling
PCPG	Per-core power gating
PAM	Power-aware mapping
MOC	Multi-objective controller
FC	Frequency cutting
RA	Reliability-aware (based of target reliability)
RS	Region selection
TM	Task mapping

reliability-aware units on the host machine. The Reliability Monitor is a process that computes Equation (5) for the node every sampling period, thus presenting a constant computational complexity on a given architecture; similar consideration can be drawn for the Reliability-aware VF capping unit. Then, the Reliability-aware RTM unit has been obtained by enhancing two state-of-the-art algorithms, without changing their original computational complexity. As evaluated in [7], [33], they have been designed for an online employment. Indeed, their execution times on a host machine, constituted by a modern desktop computer, would be in the range of some microseconds.

6 Experimental Results

A SystemC system-level simulation environment has been employed to perform an experimental evaluation of the proposed approach. The architectural model is based on Noxim NoC simulator [34] and the processing node was characterized according to the Niagara2 in-order processor specifications from McPAT [35]. Physical scaling parameters and other characteristics such as power modeling and TDP were gathered from Lumos [36]. Hotspot [37] was integrated in the simulator for the steady-state thermal model. Moreover, the considered aging mechanism has been EM, characterized as in [20]. Similarly to [8], [18], in some of the experiments, we considered also process variation to analyze its effects on the aging and the capability of the proposed reliability-aware approach to deal with this additional issue; in particular, we modeled per-core maximum frequency variation map as done also in [18] by means of a normal distribution. For the experiments, we characterized a realistic 12×12 architecture with a squared floorplan, a chip area of $138\ mm^2$ in $16\ nm$ technology, and TDP of 90 W. We set $t_{target} = 10$ years and a per-node $R(t_{target}) = 45\%$. Such values have been inspired by the curve in Fig. 3a characterized by a $60°C$ steady-state temperature, where, $R(\mathrm{MTTF}) \cong 45\%$ and MTTF ≅10 years; indeed we consider this a challenging scenario to demonstrate our approach.

TGG [38] has been used to generate workload applications. The workload has been defined as a random sequence of applications, kept fixed in all experiments for the sake of fair comparison. To evaluate our proposed strategy in run-time mapping, we performed both long-term simulations and short-term ones. Long-time simulations were used to analyze the evolution of the lifetime reliability of the various cores for the overall service life, equal to 10 years; we enlarged the execution times of the applications to last a few days to perform an accelerated experiment (with a reasonable simulation time). According to [39], the workload in night time is half of the one at day time. Thus, we divided each day into two equal parts, i.e., day time and night time, such that workload in the former is twice that of the latter. Finally, in short-time simulations, we run for approximately 3 hours to study the effects of the proposed RRM scheme on the performance of the system.

We compared our approach against

i) the nominal layer composed of MapPro [7] and CoNA [33] (namely *without Rel.*),
ii) a rotation-based spare unit strategy [29] integrated with the considered nominal approach (*rotation*), and
iii) the approach in [32] (dubbed as *Sun*), a technique to balance the reliability by defining multiple VF zones on the chip and corresponding task allocation.

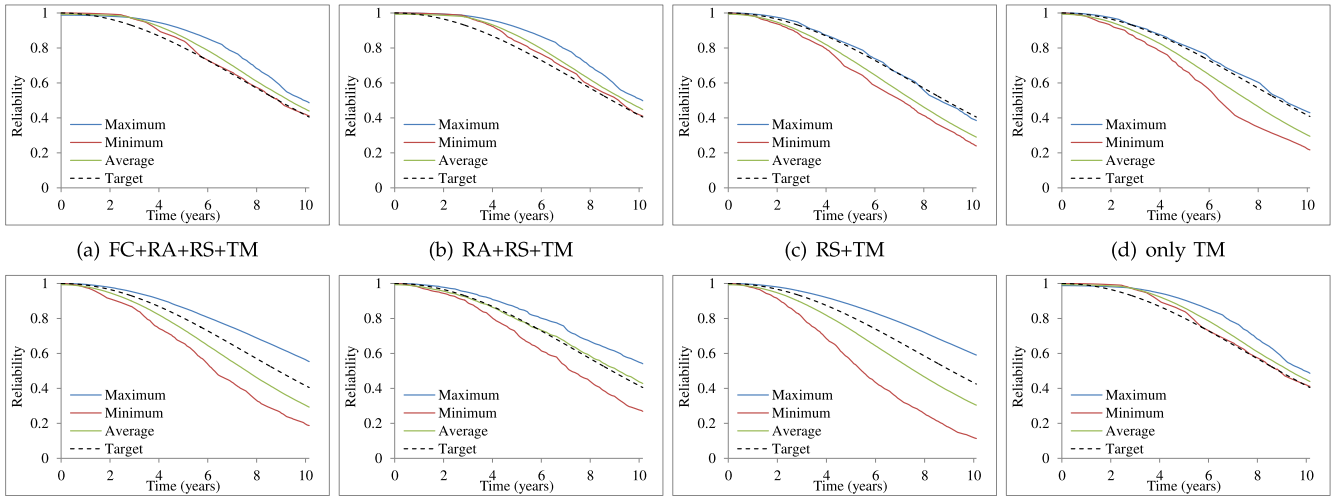
We selected these specific past works since they are the closest ones to the proposed approach based on the considered scenario, models, and optimization goal. Moreover, they were re-adapted and aligned to the proposed approach to consider same metrics and objective functions for a fair comparison; in particular, they have been adapted to use $\Delta R(t)$ as reliability metric. Moreover, we decomposed the proposed approach in various units to better evaluate the effectiveness of each contribution:

i) region selection based on Algorithm 1 (*RS*),
ii) task mapping based on Algorithm 2 (*TM*),
iii) long-term reliability awareness based on $\Delta R(t)$ metric in Lines 9–12 of Algorithm 2 (*RA*), and
iv) VF capping based on Algorithm 3 (*VC*); when *RA* basic $R(t)$ formula is used.

Finally, to show the flexibility of the proposed approach and its applicability to various power management strategies, in our experiment we considered two different state-of-the-art techniques: i) power-aware mapping approach (PAM) [40], which acts only on PCPG, and ii) multi-objective power controller approach (MOC) [11], exploiting both PCPG and DVFS to provide a more advanced and finer-grained control. To help the reader, Table 1 summarizes all the acronyms.

In the first long-term experimental campaign we compare different reliability-aware approaches on the system lifetime while PAM is used for power management. In PAM, based on the power feedback from the system and a power estimation of the most recently coming application, mapping the application is postponed until the summation of instantaneous power and estimated power of the new application is below TDP. Fig. 6 shows the reliability curves for the various considered approaches; each graph reports the minimum, the maximum and the average reliability values of the various cores within the architecture against the target reliability curve. From the achieved results we can draw the following considerations:

i) The nominal approach (without Rel.) confirms to be reliability agnostic and therefore it fails in satisfying the reliability target.
ii) The full-fledged approach (FC+RA+RS+TM) is able to accurately exploit the reliability budget and satisfy the target lifetime requirement.
iii) Also RA+RS+TM obtains similar results to the previous ones; this is because when using PAM, VF capping presents a limited effectiveness. This comes



(a) FC+RA+RS+TM (b) RA+RS+TM (c) RS+TM (d) only TM

(e) only RS (f) rotation (g) without Rel. (h) FC+RA+RS+TM while process variation

Fig. 6. Overall system reliability for the various resource management approaches with PAM.

from the fact that DVFS is not supported in PAM and, consequently, all cores are operating at their highest frequency leading to larger amount of dark areas; therefore, this gives more capability to resource management unit to balance the reliability alone regardless of VF capping process.

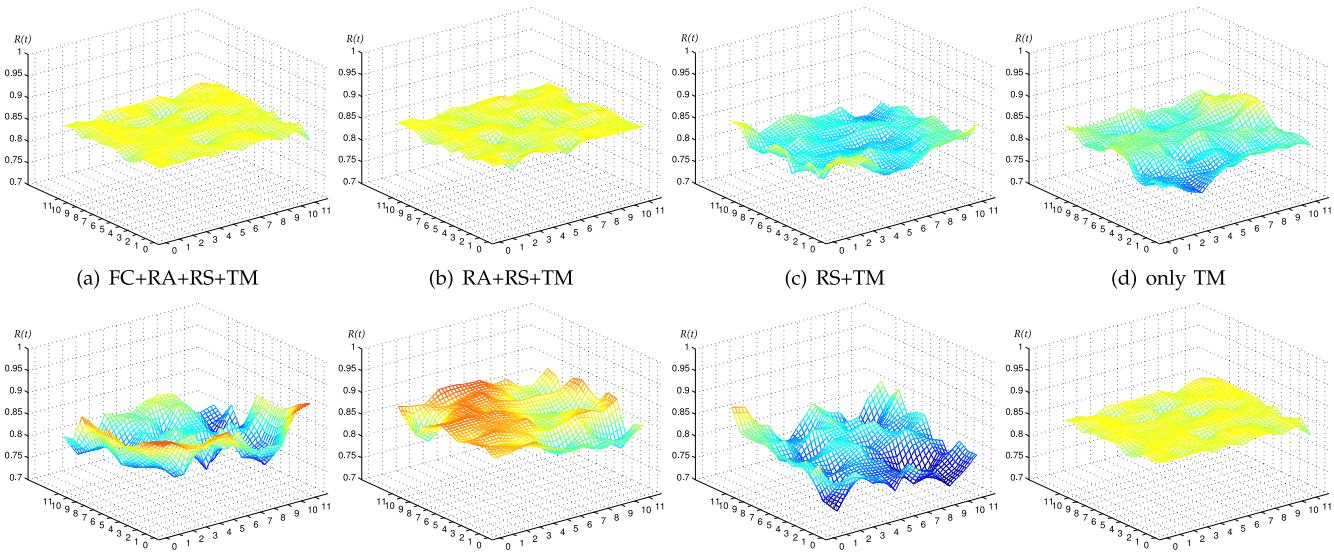
iv) When considering a subset of the units of the proposed reliability-aware approach (in graphs named RS+TM, only TM, and only RS) we obtain worse results, thus confirming the relevance of all the various contributions to the fulfilling of the reliability target.

v) The considered past approach, *rotation*, is not able to guarantee reliability requirements, because it works separately from the nominal RRM layer; therefore, differently from the multi-core scenario where the strategy was originally employed, its periodic disabling mechanism is conflicting with the necessities of the RRM layer in the application mapping, thus causing suboptimal performance results (Sun was not considered in this experimental session since it is based on DVFS while PAM is not).

vi) From Fig. 6 it is possible to state that thanks to the direct feedback loop considering the reliability metric, the full-fledged approach is able to satisfy the requirements in the case process variation is considered in the architecture.

These curves present more fluctuations than the ones in Fig. 6, since the architecture presents an heterogeneity in terms of core operating frequencies; thus, balancing reliability while providing high performance is more challenging.

To better demonstrate our claim, we report in Fig. 7 the reliability distribution on the chip for these eight scenarios after 4 years of activity. As can be seen, the reliability distribution achieved by FC+RA+RS+TM is more evenly distributed compared to all the other scenarios, while the worst case in obviously the nominal approach.



(a) FC+RA+RS+TM (b) RA+RS+TM (c) RS+TM (d) only TM

(e) only RS (f) rotation (g) without Rel. (h) FC+RA+RS+TM while process variation

Fig. 7. Reliability distribution for the various resource management strategies with PAM after 4 years of activity.

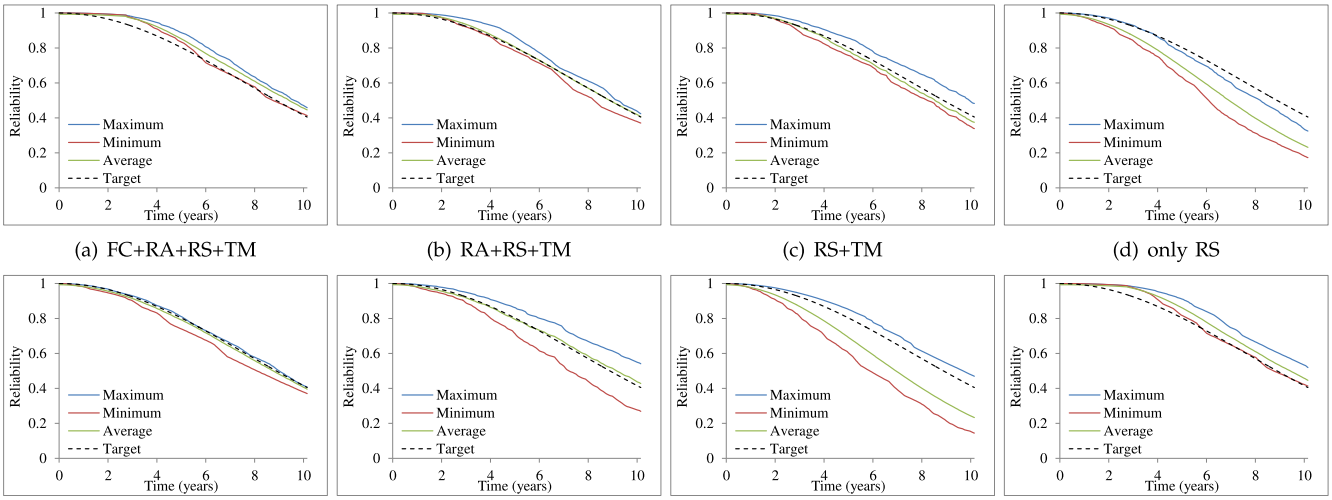


Fig. 8. Overall system reliability for the various resource management approaches with MOC.

In the second long-term experimental campaign we perform similar experiments by using MOC [11] as a power management strategy. MOC supports both PCPG and DVFS; this last one enables dim silicon [36] as discussed in Section 5.4, and, therefore, it introduces further challenges in reliability management. The results shown in Fig. 8 lead to the following considerations:

i) The overall trend presented in the previous campaign is confirmed, and, in particular, the effectiveness of the proposed full-fledged approach and the worsening when some units are disabled (RA+RS +TM, RS+TM and only RS), while the nominal approach has serious reliability issues.
ii) Differently from the previous experiment, the VF capping strategy is necessary to satisfy the reliability requirement; the motivation is related to the fact that MOC uses DVFS and therefore causes a disturbance in the reliability balancing.
iii) Both Sun and rotation approaches are not able to meet the requirements.
iv) The effectiveness of the proposed approach when process variation is considered is confirmed.

Finally, we perform short-term simulations to show the impact of the proposed approach on the overall system throughput. Fig. 9 report system throughput (in applications/hour) when using PAM and MOC, respectively.

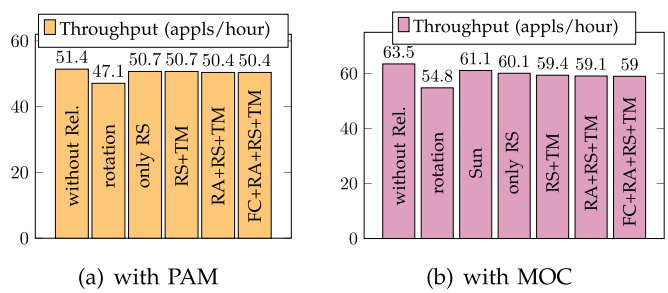


Fig. 9. System throughput for the various resource management strategies.

These results confirm that the overall system throughput penalty while applying our reliability-aware approach is limited; 2 percent performance slowdown with PAM and 7 percent with MOC. The higher penalty while using MOC in contrast with PAM is due to the fact that using DVFS decreases the dark areas resulting less freedom for mapper to fulfill both performance and reliability.

7 Conclusions

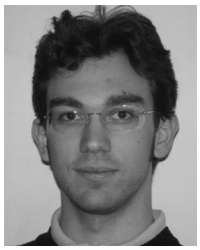
This paper has presented a novel reliability-performance co-management approach for many-core systems in dark silicon regime. The proposed approach enhances the nominal RRM layer in such architectures with a set of new units analyzing the lifetime reliability of the various cores against to a lifetime target reference, and by extending the classical mapping unit to consider reliability together with further performance-related metrics; moreover, a specific reliability-aware unit performs core-level maximum VF capping to recover from excessive stress. Experimental results demonstrated the effectiveness of the strategy to fulfill the target reliability in long term with a negligible penalty on performance in applications' execution and by not violating power budget, differently from most recent reliability-aware counterparts.

References

[1] A. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen, *Dark Side of Silicon*, 1st ed. Cham, Switzerland: Springer, 2016.
[2] Semiconductor Industry Association, International Technology Roadmap for Semiconductors. (2015). [Online]. Available: http://www.itrs2.net/
[3] JEDEC Solid State Tech. Ass., *Failure Mechanisms and Models for Semiconductor Devices*, JEDEC Standard JEP122G, 2010. [Online]. Available: https://www.jedec.org/
[4] Y. Xiang, T. Chantem, R. Dick, X. Hu, and L. Shang, "System-level reliability modeling for MPSoCs," in *Proc. Conf. Hardware/Software Codesign Syst. Synthesis*, 2010, pp. 297–306.
[5] E. Carvalho, N. Calazans, and F. Moraes, "Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs," in *Proc. Int. Workshop Rapid Syst. Prototyping*, 2007, pp. 34–40.
[6] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *Proc. Des. Autom. Conf.*, 2013, pp. 1–6.

[7] M.-H. Haghbayan, A. Kanduri, A.-M. Rahmani, P. Liljeberg, A. Jantsch, and H. Tenhunen, "MapPro: Proactive runtime mapping for dynamic workloads by quantifying ripple effect of applications on networks-on-chip," in *Proc. Int. Symp. Netw.-on-Chip*, 2015, pp. 1–8.
[8] D. Gnad, M. Shafique, F. Kriebel, S. Rehman, D. Sun, and J. Henkel, "Hayat: Harnessing dark silicon and variability for aging deceleration and balancing," in *Proc. Des. Autom. Conf.*, 2015, pp. 180:1–180:6.
[9] T. Kim, X. Huang, H. B. Chen, V. Sukharev, and S. X. D. Tan, "Learning-based dynamic reliability management for dark silicon processor considering EM effects," in *Proc. Conf. Des. Autom. Test Europe*, 2016, pp. 463–468.
[10] M. H. Haghbayan, A. Miele, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "A lifetime-aware runtime mapping approach for many-core systems in the dark silicon era," in *Proc. Conf. Des. Autom. Test Europe*, 2016, pp. 854–857.
[11] A. Rahmani, M. Haghbayan, A. Miele, P. Liljeberg, A. Janthsch, and H. Tenhunen, "Reliability-aware runtime power management for many-core systems the in dark silicon era," *IEEE Trans. on VLSI Syst.*, vol. 25, no. 2, pp. 427–440, Feb. 2017.
[12] R. van der Wijngaart, T. Mattson, andW. Haas, "Light-weight communications on Intel's single-chip cloud computer processor," *SIGOPS Oper. Syst. Rev.*, vol. 45, no. 1, pp. 73–83, 2011.
[13] Kalray, Kalray MPPA Manycore. (2016). [Online]. Available: http://www.kalrayinc.com/
[14] Adapteva, Adapteva Epiphany. (2016). [Online]. Available: http://www.adapteva.com/
[15] C. Silvano, G. Palermo, S. Xydis, and I. Stamelakos, "Voltage island management in near threshold manycore architectures to mitigate dark silicon," in *Proc. Int. Conf. Des. Autom. Test Europe*, 2014, pp. 1–6.
[16] J. Blome, S. Feng, S. Gupta, and S. Mahlke, "Self-calibrating online wearout detection," in *Proc. Int. Symp. Microarchitecture*, 2007, pp. 109–122.
[17] T. Kim, B. Zheng, H.-B. Chen, Q. Zhu, V. Sukharev, and S. X.-D. Tan, "Lifetime optimization for real-time embedded systems considering electromigration effects," in *Proc. Int. Conf. Comput.-Aided Des.*, 2014, pp. 434–439.
[18] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. Rosing, "Dynamic variability management in mobile multicore processors under lifetime constraints," in *Proc. Int. Conf. Comput. Des.*, 2014, pp. 448–455.
[19] A. Hartman and D. Thomas, "Lifetime improvement through runtime wear-based task mapping," in *Proc. Int. Conf. Hardware/Software Codesign Syst. Synthesis*, 2012, pp. 13–22.
[20] C. Bolchini, M. Carminati, M. Gribaudo, and A. Miele, "A lightweight and open-source framework for the lifetime estimation of multicore systems," in *Proc. Int. Conf. Comput. Des.*, 2014, pp. 166–172.
[21] T. Chantem, Y. Xiang, X. Hu, and R. Dick, "Enhancing multicore reliability through wear compensation in online assignment and scheduling," in *Proc. Conf. Des. Autom. Test Europe*, 2013, pp. 1373–1378.
[22] J. Srinivasan, S. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *Proc. Int. Symp. Comput. Architec.*, 2004, pp. 276–287.
[23] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Multi-Mechanism reliability modeling and management in dynamic systems," *IEEE Trans. VLSI Syst.*, vol. 16, no. 4, pp. 476–487, 2008.
[24] A. Das, A. Kumar, B. Veeravalli, C. Bolchini, and A. Miele, "Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs," in *Proc. Conf. Des. Autom. Test Europe*, 2014, pp. 61:1–61:6.
[25] ReliaSoft, ReliaSoft's reliability edge newsletter. (2000). [Online]. Available: http://www.reliasoft.com/newsletter/2Q2000/mttf.htm, Volume 1, Issue 1.
[26] A. Coskun, R. Strong, D. Tullsen, and T. S. Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proc. Int. Conf. Meas. Model. Comput. Syst.*, 2009, pp. 169–180.
[27] K. Ma and X. Wang, "PGCapping: Exploiting power gating for power capping and core lifetime balancing in CMPs," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, 2012, pp. 13–22.
[28] U. R. Karpuzcu, B. Greskamp, and J. Torrellas, "The BubbleWrap many-core: Popping cores for sequential acceleration," in *Proc. Int. Symp. Microarchitecture*, 2009, pp. 447–458.
[29] L. Huang and Q. Xu, "Characterizing the lifetime reliability of manycore processors with core-level redundancy," in *Proc. Int. Conf. Comput.-Aided Des.*, 2010, pp. 680–685.
[30] A. Y. Yamamoto and C. Ababei, "Unified reliability estimation and management of NoC based chip multiprocessors," *Microprocessors Microsystems*, vol. 38, no. 1, pp. 53–63, 2014.
[31] C. Bolchini, M. Carminati, A. Miele, A. Das, A. Kumar, and B. Veeravalli, "Run-time mapping for reliable many-cores based on energy/performance trade-offs," in *Proc. Symp. Defect Fault Tolerance VLSI Nanotech. Syst.*, 2013, pp. 58–64.
[32] J. Sun, R. Lysecky, K. Shankar, A. Kodi, A. Louri, and J. Roveda, "Workload assignment considering NBTI degradation in multicore systems," *J. Emerging Technol. Comput. Syst.*, vol. 10, no. 1, pp. 4:1–4:22, Jan. 2014.
[33] M. Fattah, M. Ramirez, M. Daneshtalab, P. Liljeberg, and J. Plosila, "CoNA: Dynamic application mapping for congestion reduction in many-core systems," in *Proc. Int. Conf. Comput. Des.*, 2012, pp. 364–370.
[34] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *Proc. Int. Conf. Appl.-Specific Syst. Architec. Processors*, 2015, pp. 162–163.
[35] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. Int. Symp. Microarchitecture*, 2009, pp. 469–480.
[36] L. Wang and K. Skadron, "Dark versus dim silicon and near-threshold computing extended results," Dept. Comput. Sci., Univ. Virginia, Charlottesville, VA, USA, Tech. Rep. TR-2013-01, 2012.
[37] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optimization*, vol. 1, pp. 94–125, 2004.
[38] TGG: Task Graph Generator. (2013). [Online]. Available: http://sourceforge.net/projects/taskgraphgen/
[39] H. Liu, "A measurement study of server utilization in public clouds," in *Proc. Int. Conf. Dependable Autonomic Secure Comput.*, 2011, pp. 435–442.
[40] M.-H. Haghbayan, et al., "Dark silicon aware power management for manycore systems under dynamic workloads," in *Proc. Int. Conf. Comput. Des.*, 2014, pp. 509–512.

Mohammad-Hashem Haghbayan received the BA degree in computer engineering from Ferdowsi University of Mashhad and the MS degree in computer architecture from the University of Tehran, Iran. Since 2014 he is working toward the PhD degree at the University of Turku, Finland. His research interests include high-performance energy-efficient architectures, software-hardware microarchitecture interaction, power management techniques, and online/offline testing. He has several years of experience working in industry and designing IP cores as well as developing research tools before starting his PhD. He is a student member of the IEEE.



Antonio Miele received the MSc degree in computer engineering from Politecnico di Milano, and the MSc degree in computer science from the University of Illinois, Chicago and the PhD degree in information technology from Politecnico di Milano, in 2010. He is an assistant professor at Politecnico di Milano since 2014. His main research interests include related to the definition of design methodologies for embedded systems, in particular focusing fault tolerance and reliability issues, runtime resource management in heterogeneous multi-/many-core systems and FPGA-based systems design. He is a member of the IEEE.

Amir M. Rahmani received the master's degree from the Department of ECE, University of Tehran, Iran, in 2009, the PhD degree from the Department of IT, University of Turku, Finland, in 2012, and the MBA degree jointly from the Turku School of Economics and European Institute of Innovation & Technology (EIT) ICT Labs, in 2014. He is EU Marie Curie Global Fellow with the University of California Irvine and TU Wien (Austria). He is also an adjunct professor (Docent) in embedded parallel and distributed computing with the University of Turku, Finland. His research interests span Self-aware Computing, runtime resource management, Healthcare Internet of Things, and fog computing. He is a member of the IEEE.

Pasi Liljeberg received the MSc and PhD degrees in electronics and information technology from the University of Turku, Turku, Finland, in 1999 and 2005, respectively. He is currently a full professor in the Embedded Electronics Laboratory. During the period 2007-2009, he held an Academy of Finland researcher position. He is the author of more than 250 peer-reviewed publications, has supervised nine PhD theses. He is the leader of the Internet-of-Things for Healthcare (IoT4Health) Research Group. He is a member of the IEEE.

Hannu Tenhunen received the diplomas from the Helsinki University of Technology, Finland, 1982, and the PhD degree from Cornell University, Ithaca, New York, 1986. In 1985, he joined the Signal Processing Laboratory, Tampere University of Technology, Finland, as an associate professor and later served as a professor and department director. Since 1992, ha has been a professor in the Royal Institute of Technology (KTH), Sweden, where he also served as a dean. He has more than 600 reviewed publications and 16 patents internationality. He is a member of the IEEE.