A Lifetime-Aware Runtime Mapping Approach for Many-core Systems in the Dark Silicon Era

Mohammad-Hashem Haghbayan[1], Antonio Miele[3], Amir M. Rahmani[1,2], Pasi Liljeberg[1], and Hannu Tenhunen[1,2]
[1]Department of Information Technology, University of Turku, Finland
[2]Department of Industrial and Medical Electronics, KTH Royal Institute of Technology, Sweden
[3]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
E-mail: mohhag@utu.fi, antonio.miele@polimi.it, amirah@utu.fi, pakrli@utu.fi, hannu@kth.se

***Abstract*—In this paper, we propose a novel lifetime reliability-aware resource management approach for many-core architectures. The approach is based on hierarchical architecture, composed of a long-term runtime reliability analysis unit and a short-term runtime mapping unit. The former periodically analyses the aging status of the various processing units with respect to a target value specified by the designer, and performs recovery actions on highly stressed cores. The calculated reliability metrics are utilized in runtime mapping of the newly arrived applications to maximize the performance of the system while fulfilling reliability requirements and the available power budget. Our extensive experimental results reveal that the proposed reliability-aware approach can efficiently select the processing cores to be used over time in order to enhance the reliability at the end of the operational life (up to 62%) while offering the comparable performance level of the state-of-the-art runtime mapping approach.**

Keywords*—*Dark Silicon, Lifetime Reliability, Many-core Architectures, Mapping, Runtime Management

I. Introduction

Ongoing technology scaling in chip production has brought massive computation parallelization with many cores fabricated on the same chip area. At the same time, this technological trend has increased power densities and, consequently, operating temperatures in components. As a result, the susceptibility of the devices to aging and wear-out phenomena (such as time dependent dielectric breakdown, thermal cycling, or electromigration) has worsened [1], thus dramatically decreasing the system lifetime. Therefore, design for reliability is becoming a necessity in modern many-core systems.

On the other hand, power has become a first-class constraint in recently fabricated many-core systems due to thermal and power constraints and the emergence of the dark silicon era [2]. Dark silicon denotes the phenomenon that, due to thermal and power constraints, the fraction of transistors that can operate at full frequency is decreasing with each technology generation. Accordingly, many efforts have been made to manage system power consumption at runtime alongside minimizing the effect of dark silicon [3], [4]. However, the dynamic nature (i.e., heterogeneity) of workloads makes the amount of dark area on the chip (i.e., total chip utilization) highly changeful that results in significant temperature gradient on the substrate leading to unbalanced cores' aging [5]. Furthermore, a stress-agnostic workload distribution approach may lead to an unbalanced aging among the cores of the system thus causing a reduction of the lifetime of the overall system. *Therefore, a paradigm shift from the conventional performance-centric runtime resource management to performance-reliability co-management is inevitable specially for many-core systems using runtime mapping techniques*.

Runtime application mapping policy is one of the key factors that determines the performance and energy efficiency of many-core systems [6] running a dynamic workload characterized by applications starting and ending dynamically with an unknown trend. Most of the existing runtime application mapping algorithms focus on communication minimization in the mapping algorithm and contiguity of the applications in the system. Moreover, with dark silicon, not all the cores can be active at the same time and the number of cores that can be active is dynamic and subjective to activity of other worker cores. This brings new opportunities to exploit dark silicon for the sake of reliability by considering the aging status of cores in runtime mapping as well as availability of the cores and power budget.

In this paper, we propose a runtime reliability-aware resource management technique for many-core systems to evenly distribute the workload stress across the system cores. The proposed technique is composed of two phases (i.e. nested feed-back controllers): i) runtime reliability analysis (i.e., the long-term management), and ii) reliability-aware runtime mapping (i.e., the short-term management). In the long term phase, a reliability analysis unit monitors architectural temperatures and computes the aging status of the system by means of a state-of-the-art statistical reliability model. In the short term phase, based on the information obtained from the long term analysis, online resource management is adopted not only to fulfill the design-time reliability requirements, but also to balance the lifetime of the system by avoiding to choose the more stressed cores. Even though, it has been generally claimed that reliability is one of the crucial challenges for many-core systems in the dark silicon era [7], to the best of our knowledge, ours is the first work to comprehensively address the lifetime reliability and aging issues while maintaining system performance at system level in the considered context. To summarize, the key contributions of this work are as follows:

- Proposing a lifetime reliability aware runtime mapping to fulfill systems' target reliability requirements while considering performance and limited power budget in many-core systems.
- Exploiting dark silicon to maximize the overall system lifetime by choosing the less stressed resources and providing long term recovery period for highly stressed cores.
- Utilizing fine-grained temperature feedback to dynamically analyze the reliability and develop design time target reliability analysis to be used as a metric in the runtime mapping algorithm.

The paper is organized as follows: Section II briefly discusses the working scenario and motivating example. Related work about dynamic mapping and reliability-aware resources allocation are covered in Section III. The proposed approach is presented in details in Section IV. Experimental results are provided and discussed in Section V and, finally, Section VI concludes the paper and discusses future work.

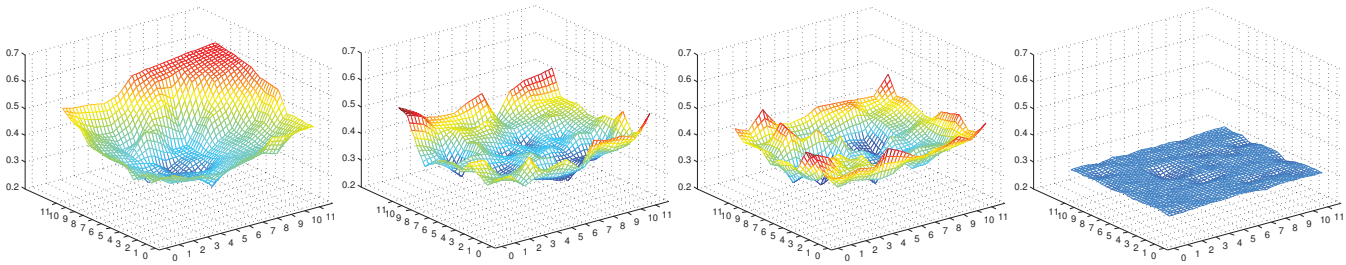
II. Working Scenario and Motivation

To demonstrate the need for performance-reliability co-management, let us consider a many-core system organized in *mesh-grid of homogeneous* cores interconnected by a Network-on-Chip (NoC) (as in [8]). Such system is generally employed in the acceleration of highly-parallel data-intensive applications organized in terms of a pipelined dataflow and represented by means of *task graphs*.

In many situations, when a system is deployed in specific appliance, the customers expect it to work for a planned service life, expressed in a number of years. Thus, we may use the classical stochastic reliability model [1], [12] to estimate the probability that the system will survive until the specified lifetime:

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

This joint work has been partially supported by the EU COST Action IC1103 - MEDIAN - ManufacturablE and Dependable multIcore Architectures at Nanoscale.



(a) Using NN [9] (life time=5.2 years) (b) Using SHiC [10] (life time=5.54 years) (c) Using MapPro [11] (lifetime=5.52 years) (d) Using reliability aware mapping (life time=12 years)

Figure 1: The effect of different runtime mapping approached on overall system lifetime (until the first core's reliability reaches 30%)

In this work, we use the extended model presented in [12] since it is able to consider aging effects due to temperature variations. When considering such model, there are two possible approaches to express the expected lifetime and to measure system reliability during its operational life: 1) to compute the definition of Mean Time To Failure (MTTF) (as in [13], [14]), or 2) to express a minimum reliability level R_{target} the system must fulfill at the end of the service life t_{target} (as in [15], [16]). We here consider the second method since in the considered highly dynamic scenario, it is more suitable for monitoring the actual system's reliability against the target value during the operational life.[1]

Now, let us assume that the target platform is a 12×12 many-core system (subject to a 30% of dark silicon on average due to power constraints) with the requirement of guaranteeing a service life of about 12 years. More precisely, similar to [15], our goal is to provide at least a 30% reliability for each single core at the end of the target lifetime. We analyzed the most relevant state-of-the-art dynamic mapping strategies for many-core systems [9]–[11] in terms of the effects of the dispatching decisions on the reliability of each single cores. It should be noted that all these state-of-the-art methods are performance-centric and reliability-agnostic. As shown in the first three plots in Figure 1, none of the approaches is able to guarantee the target. In fact, they miss the requirement before 6 years when the first core has a reliability below the 30%. This is due to the fact 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 as shown in the plots.

In this paper, we argue that by performing a reliability-aware dynamic resource allocation, it is possible to guarantee the specified reliability constraint (as shown in Figure 1(d) where only after 12 years the first core has a reliability lower than 30% thanks to the accurate stress balancing among the cores), while maintaining system performance (in terms of completed applications per time), achieving comparable results w.r.t. the reliability-agnostic approaches. In short, we perform reliability-performance co-management.

III. Related Work

Dynamic Reliability Management (DRM) to deal with an evolving workload have been investigated by various works ([13], [14], [16]–[20]) considering different architectures (e.g. bus-based multi-core systems or NoC-based architectures) and different strategies (acting on the mapping or on the Dynamic Voltage and Frequency Scaling, DVFS). The most effective knob in the NoC-based architectures is actually the mapping but very few works [13], [14], [19], [20] have addressed this issue. In [13], three basic reliability-driven mapping strategies are evaluated in terms of MTTF by considering several subsequent core failures. However, the considered workload consists of a set of unconnected jobs, thus highly simplifying the mapping problem. The approaches in [19], [20] defines migration controllers that moves tasks from elder cores to younger ones. However, the approach is too fine-grained. In fact, since device reliability changes very slowly in time (in the order of days), therefore, a periodic migration of the workload would be necessary only for applications lasting for days or weeks because task migration has a considerable overhead on NoC-based manycore systems and may result in fragmentation of application's task across the system. Nevertheless, the approaches are based on task migration that would lead to non-optimal application performance also because they are not specifically designed to deal with dynamic changes in the workload. In [14], a reliability-aware resource management approach for application mapping is proposed. However, as it is an almost exhaustive search approach for task allocation, it is highly time consuming and lacks scalability and performance consideration for NoC-based many-core systems. Finally, a dynamic mapping approach based on a many-core is introduced in [21]. Even though the idea is interesting, it does not take into account power constraints. Moreover, the mapping policy is based on a integer linear programming formulation, thus not suitable for online execution.

As a conclusion, the literature presents approaches that not completely address the reliability issues raising in the scenario of many-core systems using runtime mapping and presenting dark silicon constraints. In particular, the performance/reliability/power trade-off has not been comprehensively investigated by accurately acting on the mapping knob. Indeed, in the considered scenario, the mapping choices are more influential than the DVFS one, since the system has a power cap that does not allow to switch on all the cores at the same time. The approach proposed here aims at addressing such aspects.

IV. Proposed System Architecture

The architecture of the proposed lifetime-aware runtime mapping approach is shown in Figure 2. The approach consists in a centralized controller, implementing a feedback control loop, and is organized in two main units (and some utility ones) being in charge of application mapping and lifetime reliability management, respectively.

The **Reliability Analysis Unit** is the *long-term controller* responsible for monitoring the aging status of the various cores. The unit computes an aging reference, that is a target reliability curve defined to reach the specified reliability requirement $R(t_{target})$ at the end of the lifetime t_{target}. This aging reference will suggest to the controller how fast each core should age in order to fulfill the given reliability requirement. Then, at predefined *long-term epochs* lasting from few days to several weeks, the unit analyzes the current reliability value of each core w.r.t. the target aging reference to compute a specific reliability metric describing the aging trend. The unit gathers the cores' aging status by a utility module, called **Reliability Monitor**.

[1] At the opposite, the MTTF computation requires to know the overall reliability curve, and, in a dynamic scenario, it highly depends on all the workload variations, and consequent temperature changes.

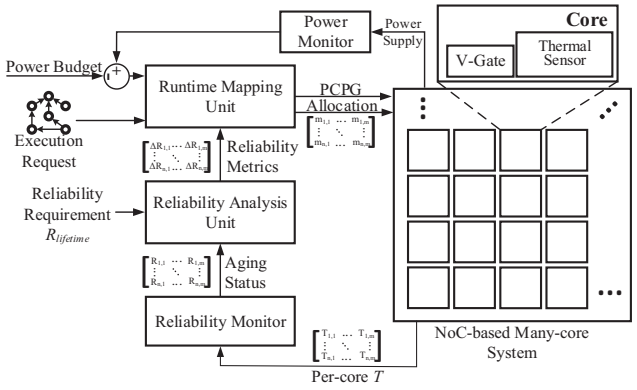


Figure 2: The overall architecture of the proposed runtime lifetime-aware mapping controller

This last one continuously updates the $R(t)$ values of the cores (every few seconds) on the basis of the temperature measures and by using the model discussed in [12]. The **Runtime Mapping Unit** is the *short-term controller* that dispatches the applications available in the incoming queue on the grid of processing cores. It takes decisions according to the profiled characteristics of the arrived application (e.g. the required dynamic power consumption), the current power consumption received by the **Power Monitor**, and the information received by the Reliability Analysis Unit.

A. The reliability Management Unit

The **Reliability Analysis Unit** monitors the cores' aging status and provides reliability metrics to the Runtime Mapping Unit. The unit defines a target reliability curve $R_{target}(t)$, called *aging reference*, on the basis of the required reliability target $R_{lifetime}$ at the given lifetime $t_{lifetime}$. This curve represents how ideally the reliability of a core working at a constant temperature will have to evolve during the lifetime to fulfill the reliability requirement. $R_{target}(t)$ is defined for all the cores as:

$$R_{target}(t) = e^{-\left(\frac{t}{\alpha_{target}}\right)^{\beta}} \quad (2)$$

where, α_{target} is obtained by inverting Equation 1:

$$\alpha_{target} = \frac{t_{lifetime}}{(-ln(R_{lifetime}))^{1/\beta}} \quad (3)$$

Then, every long-term period, the unit computes for each core $n_{w,h}$, a metric $\Delta R_{w,h}(t)$ measuring the divergence of the current reliability, received by the Reliability Monitor, and its target value:

$$\Delta R_{w,h}(t) = R_{target}(t) - R_{w,h}(t) \quad (4)$$

$\Delta R_{w,h}(t)$ represents how each core is using the reliability budget available to fulfill the overall reliability target:

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

B. Proposed Reliability-aware Mapping Approach

The aim of the Runtime Mapping Unit is to optimize system performance (in terms of number of completed applications per time), while guaranteeing reliability requirements. In fact, when considering only performance requirements, as shown in Section II, a subset of the cores would be generally preferred for applications execution, thus leading to a non-homogeneous cores' aging. At the opposite, when considering only reliability requirements, we would cause a high dispersion in application mapping, and, consequently, a considerable network congestion and low system performance. Thus, the aim of the mapping unit is to consider both the two aspects together.

To fulfill the dark silicon issues, the proposed Runtime Mapping Unit preliminarily exploits the state-of-the-art power-aware mapping (PAM) strategy from [8] to avoid thermal design power violation. it decides whether to map a new application stored in the incoming queue or wait until another running application ends, on the basis of a power prediction for the new application and the current total power consumption of the system measured through the Power Monitor. Then, the mapping of an application is performed with the classical two-stage approach: i) finding a region of most reliable cores, and then ii) mapping individual tasks in the selected region.

In the first stage, the research for the required rectangular region starts with locating a suitable *first node*. The first node represents the core around which required number of free reliable cores are available in a (almost) rectangular shape, which size depends on the number of application tasks. The size of the region is generally characterized by a radius r around the first node. This step has been automated by means of the most updated state-of-the-art strategy, called MapPro [11], that has been modified to consider at the same time reliability metrics. To introduce reliability-awareness in MapPro, we replaced the original vicinity counter metric used to identify the best region with a novel metric called *Reliability Factor* (RF). RF is defined on a square region identified by the first node $n_{w,h}$:

$$RF^{r}_{w,h}(t) = \sum_{i=i-r}^{i+r} \sum_{j=j-r}^{j+r} Wn_{i,j} \times R_{i,j}(t) \times (r - d + 1) \quad (5)$$

where $R_{i,j}(t)$ is the reliability of core $n_{i,j}$ at time t, r is the radius of selected square, and d is the distance from occupied core to the center ($r \geq d$). The formula is characterized by three factors aimed at identifying the best region in terms of reliability (we select younger cores) and achievable performance (we reduce mapping fragmentation). Wn_{ij} is the weight of core n_{ij}:

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

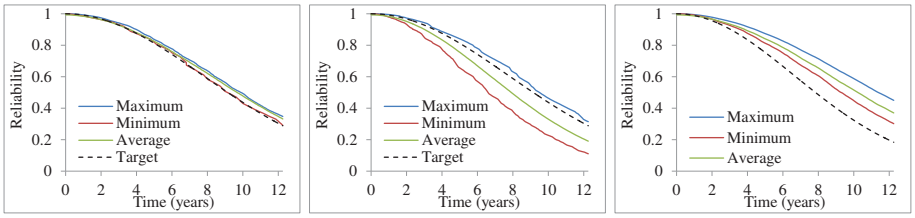
Wn_{ij} is used to discourage the selection of regions having busy cores to avoid fragmentation. Then, $R_{i,j}(t)$ factor pushes to the selection of regions with younger cores. When a core has $\Delta R_{i,j}(t) > 0$, the actual $R_{i,j}(t)$ value is replaced with 0 in the formula, to penalize the usage of such core. Finally, as discussed in [11], the last factor promotes the selection of smaller square regions.

After identifying a suitable region, the last stage maps the application tasks on the selected cores. For this stage, we used the most optimum state-of-the-art approach for inter-region mapping, that is called Contiguous Neighborhood Allocation (CoNA) [22], that was slightly modified to prioritize the usage of younger cores.

V. Experimental Results

To evaluate the proposed approach, we integrated the defined mapping policy in the simulation platform implemented in [23]. In the experiments, the architecture size is 12×12 with a squared grid floorplan, the chip area is $138mm^2$ ($16nm$ technology), and the TDP is set to 90 Watt. Then, we considered the electromigration aging mechanisms and we characterized the related parameters as in [12]. Finally, in the experiments, we assumed the required lifetime $t_{target} = 12$ years, and, to have a reasonable reliability at the end of the operational life, we set a per-core target reliability $R(t_{target}) = 30\%$. The considered workload is composed of a random sequence of applications enter the scheduler FIFO. This sequence is kept fixed in all experiments for the sake of fair comparison.

We compared our approach (namely *RM*) against i) the latest state-of-the-art TDP-based mapping approach for many-cores in dark silicon era, MapPro [11] + CoNA [22] (namely *no_RM*), and ii) a modified version of former one by adding a rotation-based spare unit strategy [13] (namely *rotation*). Moreover we performed two different experiments: 1) a coarse-grained long-time simulation, to analyze the evolution of the lifetime reliability of the various cores for the overall 12 years operational life, and 2) a fine-grained short-term simulation,



(a) RM (b) no_RM, [11] + [22] (c) rotation, [13]

Figure 3: Reliability of the system during 12 years of operational life

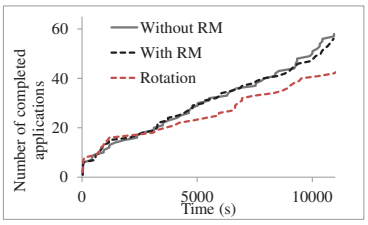


Figure 4: The number of completed applications vs. time

lasting for 3 simulated hours to study the effects of the proposed runtime mapping scheme on the system performance. Figure 3 shows the results for the first experiment. Each graph reports the minimum, the maximum and the average reliability values of the various cores within the architecture against the target reliability curve, while Figure 4 shows the performance results in terms of number of completed applications over the time.

When comparing against *no_RM*, *RM* is able to satisfy the required reliability target for the overall system by keeping the minimum core's reliability value above the specified reference curve. Moreover, it minimizes also the variance in the reliability values thus maximizing the overall lifetime of the system. Instead, since *no_RM* is reliability-agnostic, it distributes the applications without considering the aging values, and therefore lead to an unbalanced distribution of the workload and, consequently, of the aging on the cores. This will lead to a lower reliability of some cores that probabilistically will fail earlier. In fact, the reliability requirement is not satisfied since the curve representing the minimum cores' reliability is considerably below the target curve. From the graphs, it can be observed that at the end of the operational life, *RM* is able to obtain 62% improvement in the minimum reliability value w.r.t. the agnostic state-of-the-art approach. When considering the throughput results, it can be seen that *RM* is able to achieve almost the same performance of *no_RM*, i.e. the best strategy in the literature.

From this first comparison, it is possible to conclude that *RM* is able to offer the same performance of the state-of-the-art reliability-agnostic solution (i.e. *no_RM*), and at the same time to guarantee the reliability according to predefined target lifetime. This result is achieved by means of an ad-hoc selection of the processing cores guided by the reliability metrics performed over the overall lifetime. In fact, due to dark silicon issues, a limited subset of the available cores can be active at the same instant of time. *no_RM* selects square regions to be used for the mapping only by means of performance-centric metrics. However, it may happen that at the same time several areas may be equivalently used. *no_RM* selects one of them randomly. At the opposite, *RM* is able to perform an accurate selection by taking into account also the aging status. In this way, we fulfill system reliability needs and, at the same time, satisfy the other performance and power consumption requirements.

When comparing *RM* against *rotation*, this second one achieves higher reliability results. However, this aspect does not represent a benefit when considering the pursued goal, i.e., optimizing performance while fulfilling reliability requirements. In fact, *rotation* considerably reduces the performance (27% worse than *RM*) due to pursuing greedy optimizations towards reliability. This means that *RM* outperforms *rotation* since it offers higher performance by better exploiting the available reliability budget.

VI. Conclusions

A new reliability-aware mapping strategy for many-core architectures was proposed to consider lifetime reliability of a system at a runtime as well as power budget constraints and performance requirements. The proposed strategy consists of two modules: a reliability management unit, analyzing the aging status of the various cores, and a runtime mapping unit, deciding the dispatching of the applications according to performance, power and reliability needs. Experimental results show the effectiveness of the proposed approach w.r.t. the past solutions. Future work will consider heterogeneous cores, DVFS, and applications with different performance requirements.

References

[1] JEDEC Solid State Tech. Association, "Failure mechanisms and models for semiconductor devices," *JEDEC Publication JEP122G*, 2010.

[2] H. Esmaeilzadeh *et al.*, "Dark Silicon and the End of Multicore Scaling," *IEEE Micro*, vol. 32, no. 3, 2012.

[3] A.-M. Rahmani *et al.*, "Dynamic Power Management for Many-Core Platforms in the Dark Silicon Era: A Multi-Objective Control Approach," in *Proc. of ISLPED*, 2015.

[4] M. Taylor, "Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse," in *Proc. of DAC*, 2012.

[5] A. Ajami *et al.*, "Analysis of Substrate Thermal Gradient Effects on Optimal Buffer Insertion," in *Proc. of ICCAD*, 2001.

[6] C. de Souza *et al.*, "Dynamic task mapping for MPSoCs," *IEEE Design & Test of Computers*, 2010.

[7] M. Shafique *et al.*, "The EDA Challenges in the Dark Silicon Era: Temperature, Reliability, and Variability Perspectives," in *DAC*, 2014.

[8] M.-H. Haghbayan *et al.*, "Dark Silicon Aware Power Management for Manycore Systems under Dynamic Workloads," in *Proc. of ICCD*, 2014.

[9] E. Carvalho *et al.*, "Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs," in *Proc. of RSP*, 2007.

[10] M. Fattah *et al.*, "Smart hill climbing for agile dynamic mapping in many-core systems," in *Proc. of DAC*, 2013.

[11] M.-H. Haghbayan *et al.*, "MapPro: Proactive Runtime Mapping for Dynamic Workloads by Quantifying Ripple Effect of Applications on Networks-on-Chip," in *Proc. of NOCS*, 2015.

[12] C. Bolchini *et al.*, "A lightweight and open-source framework for the lifetime estimation of multicore systems," in *Proc. of ICCD*, 2014.

[13] L. Huang and Q. Xu, "Characterizing the lifetime reliability of many-core processors with core-level redundancy," in *Proc. of ICCAD*, 2010.

[14] A. Hartman and D. Thomas, "Lifetime improvement through runtime wear-based task mapping," in *Proc. of CODES+ISSS*, 2012.

[15] L. Huang and Q. Xu, "Energy-efficient task allocation and scheduling for multi-mode MPSoCs under lifetime reliability constraint," in *Proc. of DATE*, 2010.

[16] P. Mercati *et al.*, "Dynamic variability management in mobile multicore processors under lifetime constraints," in *Proc. of ICCD*, 2014.

[17] A. Tiwari and J. Torrellas, "Facelift: Hiding and Slowing Down Aging in Multicores," in *Proc. of Micro*, 2008.

[18] T. Kim *et al.*, "Lifetime Optimization for Real-time Embedded Systems Considering Electromigration Effects," in *Proc. of ICCAD*, 2014.

[19] A. Y. Yamamoto and C. Ababei, "Unified reliability estimation and management of NoC based chip multiprocessors," *Microprocessors and Microsystems*, vol. 38, no. 1, pp. 53–63, 2014.

[20] C. Bolchini *et al.*, "Run-time mapping for reliable many-cores based on energy/performance trade-offs," in *Proc. of DFT*, 2013.

[21] J. Sun *et al.*, "Workload Assignment Considering NBTI Degradation in Multicore Systems," *Journal Emerg. Technol. Comput. Syst.*, vol. 10, no. 1, pp. 4:1–4:22, Jan. 2014.

[22] M. Fattah *et al.*, "CoNA: Dynamic application mapping for congestion reduction in many-core systems," in *Proc. of ICCD*, 2012.

[23] M. Haghbayan *et al.*, "A Power-Aware Approach for Online Test Scheduling in Many-core Architectures," *IEEE Trans. on Computers*, 2016, Online Early Access.