Combined On-line Lifetime-Energy Optimization for Asymmetric Multicores

Cristiana Bolchini, Matteo Carminati
Dipartimento di Elettronica, Informazione e Bioiongegneria
Politecnico di Milano – Milano, Italy
name.surname@polimi.it

Tulika Mitra, Thannirmalai Somu Muthukaruppan
School of Computing
National University of Singapore – Singapore
{tulika|tsomu}@comp.nus.edu.sg

***Abstract*—In this paper we present an architectural and on-line resource management solution to optimize lifetime reliability of asymmetric multicores while minimizing the system energy consumption, targeting both single nodes (multicores) as well as multiple ones (cluster of multicores). The solution exploits the different characteristics of the computing resources to achieve the desired performance while optimizing the lifetime/energy trade-off. The experimental results show that a combined optimization of energy and lifetime allows for achieving an extended lifetime (similar to the one pursued by lifetime-only optimization solutions) with a marginal energy consumption detriment (less than 2%) with respect to energy-aware but aging-unaware systems.**

I. Introduction and Related Work

Over the past decade the trend in microprocessor design has shifted towards parallel architectures, such as multi/manycores, and more recently, towards asymmetric architectures, integrating cores with different power/performance characteristics, to be exploited based on the current execution context (e.g., workload, energy budget or architecture status). This flexibility introduces many possibilities as well as new challenges and issues, including an increased complexity for application mapping and resource management, to be optimized at runtime to adapt to the working conditions that may not necessarily be known in advance. Asymmetric architectures have already been proven to be beneficial in improving performance and power/energy consumption [22]. However, if the adopted policies do not take into account how they affect the system lifetime, the architecture reliability might be significantly affected. On the other hand, those solutions that focus only on lifetime improvement ([11, 15, 14, 18, 6]) are typically characterized by high energy consumption. This work proposes an approach for designing systems with combined on-line lifetime/energy optimization by exploiting the architecture asymmetry.

It may seem that the optimization of the energy consumption would naturally lead to the improvement of the system lifetime, being both aspects mainly related to temperature. However, as shown in [14], optimizing for temperature (and eventually energy) does not necessarily achieve a benefit on the lifetime reliability of the system, also considering how it is computed. In particular, as it will be shown by the experimental campaigns, the side-effects of the strategies purely devoted to the energy minimization have a negative impact on lifetime, and vice-versa.

This work proposes a resource management framework designed as an evolution of the solution presented in [22], which is focused on a power management technique for asymmetric multicores that can provide satisfactory user experience while minimizing energy consumption. We aim at building our lifetime optimization framework on the top of it, to make it aware of the components aging and exploit resource management so to extend the overall system lifetime without negatively impacting energy consumption. In particular, among all the wear-out phenomena affecting components, we explicitly target electromigration (EM) and thermal cycling (TC), even if the adopted model can be extended to integrate other effects by means of the sum of failure rates (SOFR) of wear mechanisms. The knobs used to pursue the optimization goal are the selection of the resource where to map the application and of the dynamic voltage/frequency scaling (DVFS) of the available resources.

There are a few recent approaches aimed at resource management for either power consumption and/or lifetime optimization in symmetric [1, 5, 11, 4, 14, 6] or asymmetric [10, 21, 18] architectures. The authors in [4] propose an off-line technique to improve the lifetime reliability of MPSoCs, while [11] describes a combination of design-time and runtime techniques to optimize it. It is commonly agreed upon that online adaptation is essential when dealing with aging, temperature, or energy related optimization, due to the lack of significant information that could drive a design-time solution space exploration. Thus, when moving to on-line optimization, [1] is one of the first approaches considering both lifetime reliability and energy consumption in MPSoCs; however, the two dimensions are optimized separately. Energy and reliability optimization is considered in [5] as well: the proposed hybrid approach does not consider computational energy optimization and DVFS-enabled architectures. Among recent works, [14] introduces an approach based on core utilization as a figure of merit to be balanced among homogeneous multicores, in order to improve the liferime reliability of the system. A homogeneous architecture is considered also in [6], where different techniques are evaluated and compared with the aim of improving lifetime reliability based on aging, on multicore homogeneous architectures, with no consideration for energy. Moreover, in both works, no DVFS is taken into account.

All the above-mentioned works deal with symmetric architectures. On the asymmetric side, in [21], the authors introduce dynamic reliability aware task scheduling without taking into account energy consumption optimization. Finally, among all the approaches proposed in literature, [10] is the one that

offers a solution similar to ours; however, it does not consider combined energy and lifetime reliability optimization and the reference architecture does not support DVFS, an important asset to improve energy consumption.

With respet to the current state of the art solutions, the key contributions of this work are:

- a combined lifetime/energy management framework that maximizes lifetime and minimizes energy consumption, under performance constraints. It is hierarchical, to avoid scalability issues, and lightweight, to allow on-line adaptation with a negligible overhead;
- the extension of the proposed framework from a single asymmetric node to a multi-node high performance embedded computing platform (such as [23] or [20]);
- an extensive experimental evaluation, proving the efficiency of the proposed approach based on simulations of performance and power traces collected from executions on a real ARM big.LITTLE architecture [2].

The remainder of the paper is organized as follows. Section II sets the background and defines all models for the proposed approach to be clear, while Section III presents the state of the art framework on top of which the proposed work is designed. Section IV describes the proposed runtime optimization framework, the main contribution of this work. Experimental results are shown and discussed in Section V, followed by the concluding remarks.

II. System Models

Architecture. The adopted reference architecture model is a two-level platform. At the first level, there is a grid of processing units, called *nodes*, each one considered to be an asymmetric multicore, consisting of different *clusters* composed of symmetric cores. The interconnection/communication infrastructure can be a bus, a crossbar, or a NoC, not affecting the proposed solution. Communication within a node is performed through a shared memory, allowing the applications to be seamlessly migrated across the clusters at runtime. DVFS is supported per cluster (as in [2]) therefore all the cores in a cluster run at the same frequency. The cores across the clusters are asymmetric in terms of performance, while the cores within a cluster are symmetric. We specifically target single-ISA asymmetric multicores, which exhibit power-performance heterogeneity as in ARM big.LITTLE and Tegra [17].

Applications. Serial applications only are considered in this work, possibly characterised by various *phases*. Each application i is characterized by an expected performance constraint qos_i, expressed by means of a Quality of Service (QoS) measure, for which we exploited the concept of *heartrate* [7]. It is defined as the throughput of the critical kernel in an application, such as the number of frames per second for a video encoder application. This value is provided together with the application at its arrival time T_i, assumed not to be known a-priori. The architecture must try to meet the given performance constraint, according to the soft real-time paradigm. Multiple instances of the same application and different applications are allowed to be executed simultaneously, partially or completely overlapping. However, we assume that only one application is allowed to be executed on a core at each instant of time.

Reliability. Reliability of a system or component at time t, $R(t)$, is the probability that the system has been operational up to time t. When considering wear-out phenomena, the Weibull distribution is the one that better describes the reliability of a single processor [13]:

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

where T is the processor temperature and β is the Weibull slope parameter (considered independent of T). The $\alpha(T)$ parameter formulation depends on the considered wear-out effects [13]. Here, electromigration (EM) and thermal cycling (TC) are considered. The former has an impact on the architecture wear-out that directly depends on the working temperature (high values should be avoided); the latter is negatively affected by frequent temperature variations (to be considered when dealing with DVFS). The defined model can integrate other effects either as standalone contribution or by using Sum-of-Failure Rate (SOFR) model [12] for any combination of the above failure effects. When considering EM, Black's equation [13] is used and $\alpha(T)$ is defined as:

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

where A_0 is a constant acceleration factor strongly process dependent, J is the current density, J_{crit} the critical density, E_a the activation energy, k the Boltzmann's constant, n a material-dependent constant, and $\Gamma()$ refers to the gamma function. Coffin-Manson equation [13] can be used to model TC:

$$\alpha(T)_{TC} = C_0(\Delta T - \Delta T_0)^{-q} f \tag{3}$$

where ΔT is the temperature cycling range and ΔT_0 the elastic portion of the thermal cycle (typically, $\Delta T_0 << \Delta T$, thus ΔT_0 can be dropped from Equation 3); C_0 is a material dependent constant, q is the Coffin-Manson exponent, and f is the frequency of thermal cycles. In conclusion, by exploiting the SOFR model, $\alpha(T)$ will be considered, from now on, as the sum of $\alpha(T)_{EM}$ and $\alpha(T)_{TC}$.

Mean Time To Failure (MTTF) is the adopted metric for evaluating the components lifetime. MTTF of a core is computed as the area underneath its reliability function $R(t)$: $MTTF = \int_0^{\infty} R(t)dt$. We adopt a hierarchical approach in estimating the MTTF of the entire system. The MTTF of the node is estimated by using a framework that exploits Monte Carlo simulations, random walks and the approach proposed in [11]; the MTTF of the entire system is estimated by using the first failure model, which offers a pessimistic estimation, since the system could still survive although with a lower QoS.

Energy Consumption. Two contributions are usually considered: computation and communication energy. In this work we aim at optimizing computation energy only, because of the sequential nature of the applications. Within a node, for each core, the energy consumption depends on the specific core characteristics and on the voltage levels related to the operating points it has been working at (particularly relevant when DVFS is enabled), in addition to the execution time. The energy consumption of a node is given by summing each core's contribution, and the overall energy consumption is computed by adding the contributions for all the nodes.

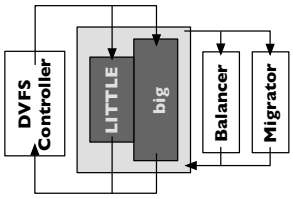

Fig. 1. Block diagram for the reference energy optimization framework.

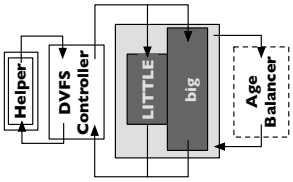

Fig. 2. Block diagram for the proposed lifetime and energy optimization framework.

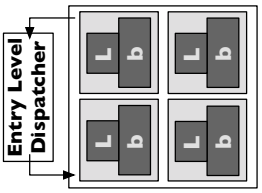

Fig. 3. Extension of the proposed framework to a mult-node architecture.

III. State of Art Framework

We here briefly introduce the architecture of the framework presented in [22], on top of which our proposal is built. The original solution manages energy consumption at the node level, where the node has an ARM big.LITTLE architecture, integrating two high performing, complex, out-of-order ARM Cortex-A15 and three energy-efficient, simple, in-order ARM Cortex-A7 cores on the same chip.

There are three relevant components for discussion. At the node level there are a *Balancer* and a *Migrator*. The former guarantees that the cores within the clusters are evenly balanced with respect to the load, by mapping incoming tasks based on the resources past utilization. The latter is in charge of migrating the applications from one cluster to the other one, when the present mapping is not the best choice in terms of energy/performance trade-off. More precisely, it moves applications that do not achieve their target QoS at the maximum frequency on the A7 cluster to the A15 cluster. Dually, it moves applications from the A15 cluster to the A7 one when the measured QoS is above the maximum target QoS at the minimum frequency in the A15 cluster.

Finally, a per-cluster *DVFS Controller* is employed. This controller manipulates the voltage-frequency levels to meet the target performance goals of all the applications in the cluster while minimizing the energy consumption. This component is based on a PID controller using a control-theoretic approach. Since the cores working points can be manipulated only at the cluster level, the target QoS of the PID controller is determined by the application with the highest computational demand. By meeting the performance target at the lowest possible voltage-frequency level, the energy consumption is minimized.

IV. Lifetime-Energy Management

The decision taken by the components of the base framework are completely aging unaware. We aim at introducing aging-aware resource management to improve the lifetime of the system with a limited impact from the energy point of view. The approach consists of two actions, one at the single node level and another one considering a multi-node scenario. While the first one is an extension of the existing architecture for incorporating the combined lifetime-energy optimization, the latter is a completely innovative solution.

A. Single Node Scenario

For the single node scenario, as shown in Figure 2, the Balancer and the Migrator have been modified and merged into a single component, the *Age Balancer*, which considers components wear-out when moving the applications between cores and clusters. A new component, the *Helper Controller*, has been introduced to support the DVFS controller decisions.

Algorithm 1 Age Balancer algorithm

```
1:  mappings = {0}
2:  clusters = getClusters()
3:  for all c in clusters do
4:      cores = getCores(clusters[c])
5:      apps = getApplications(clusters[c])
6:      for all a in apps do
7:          bestCluster = getBestCluster(apps[a])
8:          if bestCluster! = clusters[c]
            and isFree(bestCluster) then
9:              bestCluster.addApp(apps[a])
10:             apps.remove(apps[a])
11:         end if
12:     end for
13:     sortedC = coreAgeSort(cores)
14:     sortedA = appsInverseAgeSort(apps)
15:     for all i in sortedA do
16:         mappings[clusters[c]].add(sortedA[i], sortedC[i])
17:     end for
18: end for
19: return mappings
```

Age Balancer. The first mapping of each application is computed according to its worst-case QoS; the less energy-hungry resource able to satisfy the worst-case performance requirement is selected. However, the required average QoS of an application is typically much lower than the worst-case QoS, which can be gathered by looking at the applications execution traces. Thus, the initial mapping usually represents a over-provisioned design decision; for example, during phases of low computational requirements, the application could be migrated to a different cluster type and still meet the performance requirements with a much lower energy consumption. The *Age Balancer* merges the Balancer and the Migrator tasks into a single component, in charge of periodically adjusting the applications mapping, considering both their current QoS (with reference to the desired one) and the aging of the cores.

Algorithm 1 shows the pseudo-code of the *Age Balancer* heuristics. The algorithm iterates through all the clusters in the node and within each cluster, through all the applications, to find a different cluster where to move the application so that the performance requirement is satisfied and energy consumption is minimized. At the same time, within the cluster, the application that contributed the most to the aging in the previous interval is mapped onto the core aged the least. By doing this, the algorithm achieves a balanced aging across all the cores.

Helper Controller. The aim of this controller is to assist the DVFS controller in manipulating the voltage-frequency

levels of the cores; one controller for each cluster is needed, since the DVFS is available per-cluster.

As mentioned, the PID controller is in charge of guaranteeing the performance target to be met at the lowest voltage-frequency level. However, the dynamic phases of an application can make the PID controller switch aggressively between these levels. Although aggressive scaling improves energy consumption, it also has significant negative impact on the lifetime reliability through thermal cycling. To avoid this phenomenon, we propose a *Helper Controller* that prevents the drastic modification in voltage-frequency level as follows. By reading the power sensors, this controller can measure the actual power consumption of the cluster. It estimates the power consumption at different voltage-frequency levels by using equation $P = A \times C \times V^2 \times F$, where A is the activity factor, C is the load capacitance, V is the voltage and F is the frequency. The target power consumption can be estimated given the target voltage and frequency. Using the estimated power consumption, the helper controller employs the widely used thermal RC model [8] to estimate the temperature at various voltage-frequency levels. From the current temperature measurement and Equation 3, the helper controller can prune the set of voltage-frequency levels that can potentially affect the lifetime reliability of the system by a certain threshold. In our work, we assume the threshold to be 10% of the actual MTTF of the system. More precisely, to prevent TC, when the PID controller is transitioning from higher to lower frequency, the DVFS controller selects the highest values between the PID controller and the minimum of the frequencies computed by helper controller. When the transition is from lower to higher frequency, the DVFS controller selects the lowest value between the PID controller and the maximum frequency computed by the helper controller.

Invocation Frequency. In the proposed framework, the different components have to be properly coordinated and invoked at different frequencies. The per-cluster DVFS controller is invoked at a higher frequency compared to the Age balancer. There are three major reasons behind the aforementioned choice. First, the age balancer focuses primarily on improving the lifetime reliability of the system. It has been a well-established phenomenon that the lifetime reliability of a microprocessor is significantly related to the temperature. As the temperature changes occur slowly [8], the age balancer can be invoked at a much lower frequency. Second, the overhead of invoking the age balancer is high if compared to the one of DVFS controller. In ARM big.LITTLE, the overhead of migrating application across clusters is in the order of milliseconds (from 2 to 4 ms) [22]. Therefore, the age balancer has to be invoked very infrequently. Lastly, the overhead of changing voltage-frequency levels is quite minimal [22]. Changing voltage-frequency levels can positive impact the energy consumption of the system; therefore, the DVFS controller is invoked at a higher frequency.

B. Multi-Node Scenario

When moving from a single ARM big.LITTLE node architecture to a multiple node one, a new component for managing the overall architecture is introduced. The aim of this entity, dubbed *Entry Level Dispatcher* (as shown in Figure 3), is to select the best initial mapping for the incoming applications; it must take nodes aging into consideration and, if possible, select the best energy-aware solution as well. The dispatcher takes decisions based on the:

- application profiles, consisting of its worst-case required QoS and its average power consumption, for each different core type;
- system status, i.e. the number of *active* and *free* clusters/nodes.

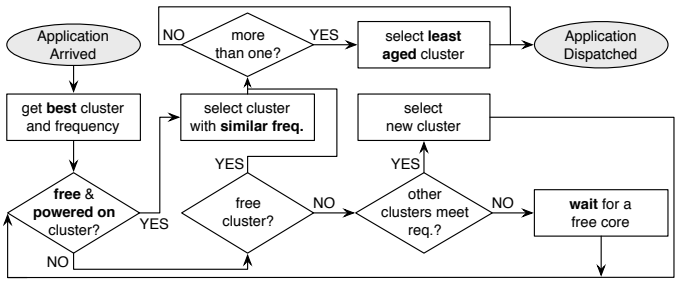

Fig. 4. Control flow chart for the dispatching algorithm.

These profiles are obtained using off-line analysis. The worst-case required QoS and average power consumption are measured at the maximum voltage-frequency level for each core type. Assuming a linear relationship between QoS and voltage-frequency level, we can estimate the performance at different frequency levels; similarly, the power consumption can be estimated as well, as already explained. We assume that the asymmetric multicore is equipped with per-core sensors measuring power, energy, voltage, frequency and temperature. We also assume that each core is equipped with wear sensors similar to the ones exploited in [10]. The dispatcher can access the sensors information to determine the system status. The dispatcher schedules the incoming applications to the appropriate node and cluster, aiming at improving the lifetime reliability while minimizing the energy, with performance as a constraint. The flow chart is shown in Figure 4.

The arriving applications are enqueued in a FIFO queue. Using the application profile, the dispatcher can estimate the best cluster type and its voltage-frequency level that can satisfy the applications performance constraint at a minimal energy consumption. First, the dispatcher identifies already powered-on clusters of the same type having free cores: these are the ideal candidates. For a lifetime reliability-aware mapping, the dispatcher selects the cluster that is least aged among the clusters with similar voltage-frequency levels. However, if the dispatcher does not find any powered-on cluster with free cores, it powers up a free cluster similar to the type previously estimated. If no such cluster is available, the dispatcher selects another cluster type that can meet the performance at the lowest possible energy cost. Last, if no free clusters are available, it waits until a core becomes free.

V. Experimental Evaluation

The experimental sessions here described aim at demonstrating how the proposed framework i) improves the results obtained by the base framework in terms of architecture lifetime without heavily impacting on energy consumption, ii) can be extended for managing the resources of a multiple

node architecture by introducing a new smart application dispatcher able to improve lifetime keeping the same (almost negligible) energy consumption overhead, and, iii) achieves almost the same lifetime improvements as a dedicated lifetime optimization framework, also improving energy consumption.

More precisely, we take as reference for our comparison two single objective frameworks optimizing energy (i.e, *Energy-only* [22]) or lifetime (a heuristic for MTTF maximization inspired by [5], *MTTF-only*). We have extended such reference policies also to the multi-node scenario although they have been designed for the single node one, to have comparative values.

A. Experimental Setup

The experimental platform is a four-node architecture, where each node is an ARM big.LITTLE. A single node is considered for the first experimental session. We collected applications power and performance traces by running applications on a real ARM big.LITTLE core using a Versatile Express development platform [2], built at $45nm$ GP technology. The test chip consists of two core Cortex-A15 (big) cluster and three core Cortex-A7 (LITTLE) cluster. Since no per-core thermal sensors are available on-board, we developed a simulation environment, based on SystemC/TLM, integrating the models of collected real data characterizing both the applications and the board. The simulator is fundamental in estimating the architecture thermal profile: it implements a steady-state temperature model, validated by means of HotSpot [8], that takes into account the self-activity of each core, the operating frequencies and the neighbor cores' temperature. The obtained temperatures are processed and used to compute the cores' reliability according to Equations 1–3. We referred to [11] for choosing the values of the parameters related to EM, and to [16] for the ones related to TC.

Real-life applications have been used to run the experiments, selected from well-known benchmarking suites, namely PARSEC [3] (*blackscholes*, *bodytrack*, *swaption*, *x264*), SPEC [19] (*h264*), and Vision [9] (*disparity*, *texture*). These applications have been selected to represent a good mix of heterogeneous behaviors. All the applications have been executed on both the A15 and on the A7 core for each of the available working frequencies; from $500MHz$ to $1.2GHz$, with a $100MHz$ step, for A15; from $600MHz$ to $1.0GHz$, with a $100MHz$ step, for A7. Most of the applications executing under $600MHz$ on A7 experienced very poor performance, therefore we discarded the related traces. For each <app, freq> pair a trace file is computed and stored, containing the application's power consumption and its performance at regular time intervals. The power sensors in ARM big.LITTLE have been used to measure power consumption. The QoS of the applications are expressed in terms of heartrate; we inserted the heartbeat register points as mentioned in [22].

Ideally, each single application has to be profiled individually to estimate the speedup for various voltage-frequency levels, resulting in complex off-line analyses. To avoid extensive profiling, we assume a linear relationship between frequency and heartrate for all the applications. The invocation frequency of age balancer is $200ms$; the DVFS controller one is $50ms$, which is much higher than the Linux scheduling epoch ($10ms$). Both frequencies have been empirically selected to obtain the best accuracy/overhead trade-off. The overhead of changing voltage-frequency levels is assumed to be $50\mu s$; application re-mapping within the same cluster takes $200\mu s$, $4ms$ across clusters [22].

B. Single-Node Evaluation

For the first execution scenario, we defined five different use cases (UC_{sx}) consisting of 50 applications, randomly selected among the applications previously presented. For each application: i) the arrival time is computed according to a Poisson distribution (described by its parameter λ) and ii) the expected heartrate is computed as the average heartrate of a randomly selected trace among the available ones. The λ value is constant and chosen such that the architecture is fully loaded with no waiting queues.

Figure 5 shows the results for this first campaign. In the top graph, we report the ratio between the computed MTTF and the MTTF achieved by the lifetime-only optimization approach (dashed horizontal line, [5]). As expected, both the energy-only and the proposed approach are not as effective, however on average they reach 80.6% and 92.3% of the *MTTF-only* lifetime, respectively. The bottom part of the graph reports the corresponding energy ratio with respect to the energy consumption of the energy-only optimization approach (dashed horizontal line, [22]). In this case, the energy overhead for *Proposed* is, on average, 1.8% higher, while the lifetime optimization framework deteriorates the best results by 25.4%.

C. Multi-Node Evaluation

As mentioned, we have extended the policies of the reference framework to the multi-node scenario in a straightforward manner, to have a baseline reference. The same application scenario has been adopted, while the value of λ has been tuned again to have the architecture constantly busy, but to avoid waiting queues. The results, plotted in Figure 6, show almost the same trend of the previous experimental session. The proposed technique proves to still be able to mimic the optimal results in terms of lifetime, achieving, on average, 90.6% of the optimal lifetime. On the other hand, the lifetime obtained by the aging unaware solution drops to 47.8%. Results for energy consumption are more similar to the previous scenario. The *Proposed* solution maintains an average 1.3% overhead with respect to the best energy consumption values, while the overhead of the energy unaware solution is 21.1%.

D. Discussion

As expected, the frameworks optimizing a single objective (energy or lifetime) excel with respect to the metric they adopt, being characterized by high impact on the other aspect; the energy-optimization framework also has some limitations in terms of scalability. On the other hand, the combined solution we propose achieves results that are more similar to the optimal ones, for both lifetime and energy consumption; it shows a negligible overhead in energy consumption and a significant lifetime extension. More precisely, we can also note that although lifetime is related to temperature as well as energy-optimization policies, it is necessary to consider lifetime reliability as a first-class citizen to actually obtain optimal results

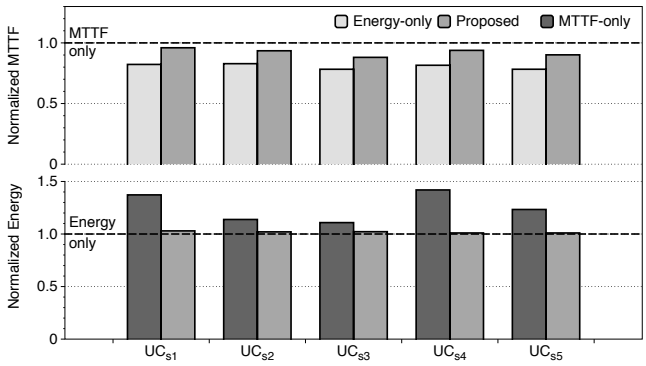

Fig. 5. The proposed approach compared with the baseline framework in a *single node* architecture.

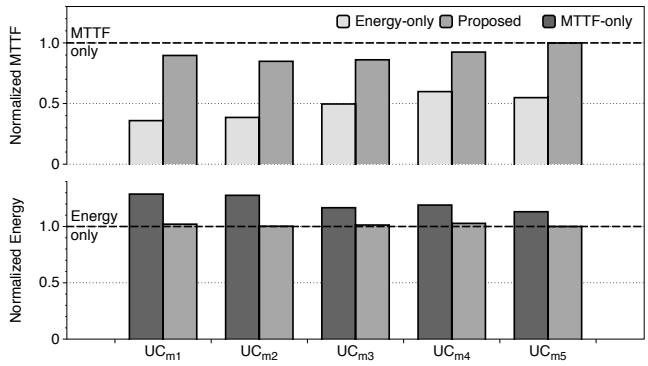

Fig. 6. The proposed approach compared with the baseline framework in a *multi node* architecture.

with respect to such aspects. Furthermore, experimental results show that the combined optimization introduces only marginal overheads on the independent metrics, marginally reducing the potential optimal condition of a single-goal optimization. Altogether, the introduction of a second relevant optimization objective has a very limited overhead with respect to the benefits it introduces.

It is also worth noting how, when moving from the single to the multi-node scenario, the combined approach shows good scalability: energy overhead from 1.8% to 1.3% and lifetime from 92.3% to 90.6% with respect to the optimization policy that focuses on one goal at the time.

VI. Conclusion

In this paper we presented the design and implementation of a framework for the combined optimization of lifetime and energy consumption in asymmetric and high performance embedded architectures. The framework is built on top of a state of art energy optimization solution and aims at making it aging-aware. Experimental results, on an architecture built with ARM big.LITTLE cores, confirm that the proposed framework, without any a-priori information, can achieve results that are comparable with those obtained by approaches that perform single objective optimization, both in the single node scenario and in the multi-node one.

Acknowledgement

This work was partially funded by the European Commission in the context of the FP7 SAVE project (#610996-SAVE) and by the Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-088.

References

[1] A.K. Coskun et al. Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors. In *Int. Conf. Measurement and Modeling Computer Systems*, pages 169–180, 2009.

[2] ARM Ltd. `http://www.arm.com/products/tools/development-boards/versatile-express/index.php`, 2011.

[3] C. Bienia et al. The PARSEC benchmark suite: Characterization and architectural implications. In *Int. Conf. Parallel Architectures and Compilation Techniques*, pages 72–81, 2008.

[4] A. Das, A. Kumar, and B. Veeravalli. Temperature aware energy-reliability trade-offs for mapping of throughput-constrained applications on multimedia MPSoCs. In *Conf. Design, Automation & Test in Europe*, pages 1–6, 2014.

[5] C. Bolchini et al. Run-time mapping for reliable many-cores based on energy/performance trade-offs. In *Symp. Defect and Fault Tolerance*, pages 58–64, 2013.

[6] C. Bolchini et al. Lifetime-aware load distribution policies in multi-core systems: an in-depth analysis. In *Conf. Design, Automation & Test in Europe*, pages 804–809, 2016.

[7] H. Hoffmann et al. Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments. In *Int. Conf. Autonomic Computing*, pages 79–88, 2010.

[8] K. Skadron et al. Temperature-aware microarchitecture: Modeling and implementation. *Trans. Architecture Code Optimization*, 1(1):94–125, 2004.

[9] S.K. Venkata et al. SD-VBS: The San Diego vision benchmark suite. In *Int. Symp. Workload Characteriz.*, pages 55–64, 2009.

[10] A.S. Hartman and D.E. Thomas. Lifetime improvement through runtime wear-based task mapping. In *Int. Conf. Hw/Sw codesign and system synthesis*, pages 13–22, 2012.

[11] L. Huang, F. Yuan, and Q. Xu. On task allocation and scheduling for lifetime extension of platform-based MPSoC designs. *Trans. Parallel Distributed Systems*, 22(12):2088–2099, 2011.

[12] J. Srinivasan et al. The case for lifetime reliability-aware microprocessors. In *Int. Symp. Comp. Arch.*, pages 276–287, 2004.

[13] JEDEC Association. Failure mechanisms and models for semiconductor devices. *JEDEC Publication JEP122G*, 2010.

[14] Yue Ma, Thidapat Chantem, X. Sharon Hu, and Robert P. Dick. Improving lifetime of multicore soft real-time systems through global utilization control. In *Great Lakes Symposium on VLSI*, pages 79–82, 2015.

[15] P. Mercati, A. Bartolini, F. Paterna, T.S. Rosing, and L. Benini. Workload and user experience-aware dynamic reliability management in multicore processors. In *Design Automation Conf.*, pages 2:1–2:6, 2013.

[16] H.V. Nguyen. Multilevel interconnect reliability: On the effects of electro-thermomechanical stresses. Ph.D. dissertation, University of Twente, Netherland. 2004.

[17] NVidia Corp. The benefits of multiple cpu cores in mobile devices., 2011.

[18] W. J. Song, S. Mukhopadhyay, and S. Yalamanchili. Amdahl?s law for lifetime reliability scaling in heterogeneous multicore processors. In *Int. Symp. High-Performance Computer Architecture*, 2016.

[19] SPEC CPU Bench. `www.spec.org/benchmarks.html`, 2014.

[20] STMicroelectronics and CEA. Platform 2012: A many-core programmable accelerator for Ultra-Efficient Embedded Computing in Nanometer Technology. In *STM Platform 2012 Workshop*, 2010.

[21] T. Chantem et al. Enhancing multicore reliability through wear compensation in online assignment and scheduling. In *Conf. Design, Automation & Test in Europe*, pages 1373–1378, 2013.

[22] T. Somu Muthukaruppan et al. Hierarchical power management for asymmetric multi-core in dark silicon era. In *Design Automation Conf.*, pages 174:1–174:9, 2013.

[23] Teraflux. Definition of ISA extensions, custom devices and External COTSon API extensions. In *Teraflux: Exploiting dataflow parallelism in Tera-device Computing*, 2011.