Dynamic Lifetime Reliability Management for Chip Multiprocessors

Milad Ghorbani Moghaddam, *Student Member, IEEE,* and Cristinel Ababei, *Senior Member, IEEE*

Abstract—We introduce an algorithm for dynamic lifetime reliability optimization of chip multiprocessors (CMPs). The proposed dynamic reliability management (DRM) algorithm combines thread migration and dynamic voltage and frequency scaling (DVFS) as the two primary techniques to change the CMP operation. The goal is to increase the lifetime reliability of the overall system to the desired target with minimal performance degradation. We test the proposed algorithm with a variety of benchmarks on 16 and 64 core network-on-chip (NoC) based CMP architectures. Full-system based simulations using a customized GEM5 simulator demonstrate that lifetime reliability can be improved by 100% for an average performance penalty of 7.7% and 8.7% for the two CMP architectures.

Index Terms—lifetime reliability; TDDB; NBTI; chip multiprocessors; network-on-chip; thread migration; DVFS

1 Introduction

Two of the most adverse wearout or aging mechanisms in deep submicron technologies include time dependent dielectric breakdown (TDDB) and negative bias temperature instability (NBTI) [1]. Their cumulative effect can cause performance degradation and eventual device and integrated circuit failure. Because one of the factors that impact TDDB and NBTI aging mechanisms is temperature [2], it is the chip multiprocessors' (CMPs) lifetime reliability that is especially affected − because their operation temperatures have been increasing due to the increased power densities. Consequently, researchers recognize that lifetime reliability is becoming a primary design concern and started to investigate methods to mitigate the negative impact of these aging effects.

Previous lifetime reliability oriented design methods fall into one of two general categories, static and dynamic approaches. Methods in the *static* category address the problem of reliability at design time. Static design methods include guardbanding and fault tolerance techniques [3], [4], [5]. Guardbanding has been used successfully to mitigate or eliminate transient, intermittent, and permanent errors. For example, supply voltages are selected high enough in order to guarantee correct functionality despite variation in threshold voltage or in temperature and supply noise. But, in this way energy gained from downscaling is sacrificed to combat potential reliability problems. However, if this sacrifice becomes too large, downscaling may become detrimental [6] and therefore such techniques may not be as effective as before.

The second category of reliability oriented design methods is that of *dynamic* approaches. The main idea of these approaches is to dynamically monitor the system during runtime and by using either *reactive* or *proactive* techniques to change the operation of the system such that lifetime reliability is improved. The proposed dynamic reliability management (DRM) algorithm falls in this category. We discuss previous works in this category in the next section. Our work combines thread migration and dynamic voltage and frequency scaling (DVFS) techniques. The proposed algorithm uses a simple yet effective heuristic approach in order to seamlessly use the two techniques to adaptively change the CMP operation such that the lifetime reliability of the overall system is increased to the desired target with minimal performance degradation. We develop and validate the proposed method by considering chip multiprocessors as systems that are formed by two crucial components: the cores as processing elements and the NoC as the communication component. The primary target application scenario of the investigated CMP architectures is datacenters, under the assumption that benchmarks like those studied in our simulations are run continuously.

2 Related Work

In this section, we discuss previous literature on reliability management in processors. Dynamic reliability banking is proposed in [7] to address aging due to electromigration (EM), another aging mechanism. A two phase DRM algorithm that chooses among predefined hardware configurations is introduced in [8]. Reliability slack is introduced in [9] and used for dynamic reliability management during periods of high processing demand. The authors of [10] exploit the variation in workloads to assign jobs to cores in a manner that minimizes the impact of NBTI and TDDB. The authors of [11] introduce Facelift, a technique to hide aging through aging-driven application scheduling and to slow it down by applying voltage changes at key times. A DVFS control and look-up table reliability estimation based DRM scheme is introduced in [12] for singlecore processors to address process variation aware oxide breakdown. The impact of job scheduling based power management on reliability is investigated in [13]. A system level HW/SW reliability management scheme where a chip dynamically adjusts its own operating frequency and supply voltage

• *M.G. Moghaddam and C. Ababei are with the Department of Electrical and Computer Engineering, Marquette University, Milwaukee WI, 53233.*
E-mail: milad.ghorbanimoghaddam@marquette.edu; cristinel.ababei@marquette.edu

Manuscript received February 13, 2018; revised February 13, 2018.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMSCS.2018.2870187, IEEE Transactions on Multi-Scale Computing Systems

over time as the devices age due to NBTI is introduced in [14]. The authors of [15] study a control theoretic approach that uses data from aging sensors to compute the wearout degradation and to maximize the lifetime of homogeneous multicore systems. A reinforcement learning algorithm is proposed in [16] to optimize the lifetime of a multicore system by controlling the average temperature and thermal cycling. The study in [17] introduces a wearout-decelerating scheme to mitigate the impact of NBTI and hot-carrier injection (HCI) in NoCs. The study in [18] presents a reliability management solution for dark silicon chips. The solution considers soft errors, process variations, and the thermal design power constraint. Simulation results were reported for 80x80 grid cells chips of LEON3 processors. This work is further extended in [19]. The same research group proposed in [20] a run-time approach that harnesses dark silicon to decelerate and balance temperature-dependent aging. Their solution also considered variability to improve the system performance for a given lifetime. They focus on NBTI and did not report if the communication among cores is via the NoC. Furthermore, the study in [21] proposed a process variation- and aging-aware dynamic hierarchical mapping solution to maximize lifetime reliability of manycore systems while satisfying performance, power, and thermal constraints. The authors reported improved system lifetime reliability by up to 2 years for 64-core and 256-core systems. To save space, for discussion of additional reliability studies we kindly refer the reader to recent survey [27].

One limitation of previous studies is that they focus separately on either the *computational* component of a processor (i.e., single core or multicore processing unit) or the *communication* component, typically the NoC. Not considering either of these components introduces significant errors, because both computational and communication units of multicore processors may become a reliability bottleneck. In our previous work [22], we found that when one does not consider for example the NoC component in the reliability optimization, the errors in MTTF values, as the most popular way to measure lifetime reliability, may be off by as much as 60%. Such errors can mislead any lifetime reliability optimization method and result into suboptimal solutions. More recently, other researchers also considered reliability of the whole chip as the combination of communication and computation components [24], [25]. The study in [25] focused on 3D NoC based CMPs, considered electromigration in power delivery network wires and BTI aging mechanisms, and proposed the ARTEMIS framework for aging-aware application mapping and DVS scheduling. The framework enables the execution of 25 percent more applications over the chip lifetime. The work in [24] addressed transient faults as well as HCI and NBTI aging mechanisms via adaptive application mapping and DVS. Their CHARM framework was reported to achieve 2.5x improvement in lifetime and up to 6x improvement in number of applications executed during the lifetime of the assumed 60 core chip.

3 Proposed Hybrid Dynamic Reliability Management

3.1 Block Diagram

The block diagram of the proposed DRM approach is shown in Fig. 1. The idea is to implement a control algorithm, which

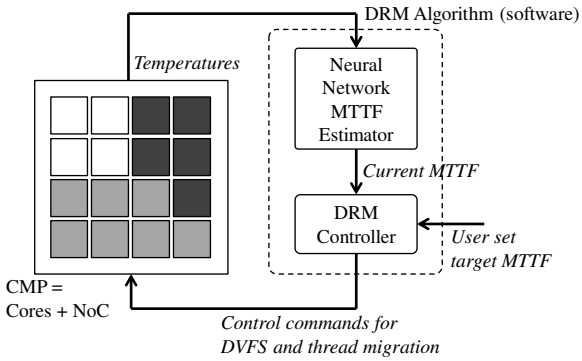


Fig. 1. Proposed dynamic reliability management algorithm has two components, the MTTF online estimator and the DRM controller. The CMP is composed of a number of tiles. A tile is (core + NoC router)

continuously monitors the temperatures of all components of both computational and communication units of the CMP hardware platform. This algorithm operates periodically according to a pre-defined *control period* and uses the input temperatures as well as the user set desired lifetime reliability target to generate output control commands that dictate how thread migration among tiles and DVFS of individual tiles is done during the next control period. These commands are generated such that lifetime reliability converges toward the desired target. The algorithm is implemented in software and has two main components. The first component estimates the current lifetime reliability and is implemented with a neural network (NN) model. Its role is to produce an estimate of the mean time to failure (MTTF) of the entire CMP as a way to quantify or measure the lifetime reliability. We use an NN model based predictor for efficiency reasons and because of NN models have been shown to provide high quality prediction and classification results. Please refer to our previous work in [22] for details on the construction of the NN model.

The second component shown in Fig. 1 is the DRM controller. Its role is to compare the currently estimated or projected MTTF to the desired target and then decide for each tile (core + NoC router) whether the clock frequency must be throttled, increased, or left unchanged or whether threads should be migrated from hot to colder tiles. The control loop from Fig. 1 shares in philosophy with any other closed-loop control theory algorithm. However, the context in which we use elements of control theory is specific in this case to the optimization of lifetime reliability for chip multiprocessors, which we handle in a unified manner, as the combination of both cores and network-on-chip. The neural-network based estimation is another specific element. The most challenging aspect of the proposed DRM algorithm is to figure out a way to make these decisions such that performance is not affected too much. The next section elaborates on how that is done.

3.2 DRM Controller

Here, we describe how we arrived to the implementation of the logic behind the DRM controller from Fig. 1. First, based on our experience from our previous studies [22], [23], we present several design insights.

We found that lifetime reliability can be more effectively improved using DVFS based techniques, but at the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMSCS.2018.2870187, IEEE Transactions on Multi-Scale Computing Systems

expense of larger performance penalties when compared to thread migration based techniques. This suggests that, for applications where performance degradation can be tolerated, DVFS based DRM schemes can be used to trade performance for larger MTTF improvements. In contrast, for applications where performance degradation is not acceptable, thread migration based DRM schemes may be a better choice. However, thread migration is limited in its ability to significantly improve MTTF even if it would be acceptable to degrade performance. That is because no matter how much one would shuffle jobs among tiles, if the application benchmark is computationally intensive and all cores are heavily utilized, temperature profile will be always high anyways. The problem is that we do not know at design time what kind of application benchmarks will be run on a given instance of a chip multiprocessor. A subtle design insight here is that the above statements about thread migration techniques are true only when the application benchmark has a number of active threads that is comparable to the total number of tiles of the CMP. That was the case in our study from [22], where the used Parsec benchmarks were specifically compiled for the duration of the so called region of interest (ROI) to run a number of threads equal to the number of cores. In such situations, a thread migration technique has little ability to improve the thermal profile via shuffling threads between tiles because all cores are already busy. However, if the application benchmark is compiled such that the number of active threads is smaller than the number of available cores, at any given time, then, thread migration can effectively be used to achieve significant reliability improvements. While this may not seem realistic or desirable because we would like all cores to do useful work at all times, applications that are programmed to be running parallel on multicore platforms may have situations when some of the tasks running on some cores finish early compared to other tasks running on other cores. Situations like that offer the opportunity to be exploited for example for thread migration. Moreover, with the increasing power dissipation on multicore platforms, researchers are already talking about the new era of dark-silicon [18], [20], [21]. That is, in dark-silicon, many cores would be shut-down or not be utilized in order to keep the total number of cores executing at the same time small enough and thus keep the temperatures low. These situations are examples when the number of threads can be less than the number of cores.

To further understand the relation between the number of available free cores (i.e., currently not having any running threads on them), which usually have colder temperatures, and the amount of MTTF improvement when thread migration is used as the main technique for lifetime reliability management, we conducted the following experiment using our modified GEM5 based simulation framework. This simulation framework will be described in more details later on. We use an NoC based CMP with 16 cores to run several benchmarks that are compiled to run during the ROI using a number of threads varied between 1 and 16. Thus we conduct 16 different full system simulations for the given benchmark. We use our own thread migration based DRM scheme [22] and the objective is to see with how much MTTF of the whole CMP system can be improved. The results of this experiment for the *blackscholes* benchmark are shown in Fig. 2.

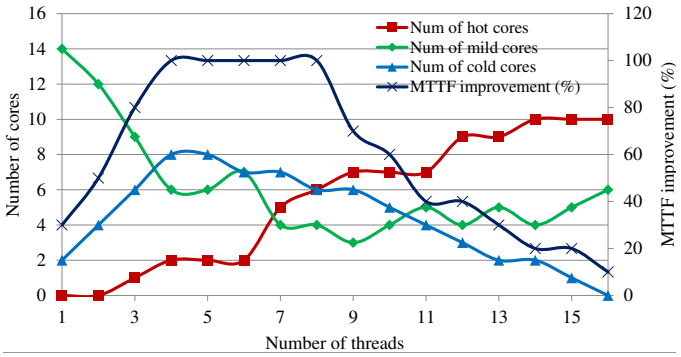


Fig. 2. Plot showing the amount of MTTF improvement using a thread migration based DRM scheme over the reference case when no DRM scheme is used at all. A tile is denoted as cold if its temperature $T < 40°C$, as mild if $40°C \leq T \leq 60°C$, and hot if $T > 60°C$.

The plot from Fig. 2 confirms the intuition that more cold tiles available provide more opportunities to the thread migration DRM scheme to migrate threads in the attempt to keep tile temperatures within a power profile that improves the MTTF. In our simulations, we found that when the number of hot cores is less than half of the total number of cores, the MTTF can be improved via thread migration to larger extents than when most of the cores are hot. This observation motivates us to implement the DRM controller as described in Fig. 3. The main idea of the controller is to 1) use as much as possible the thread migration technique because it is the cheapest to implement and provides good enough MTTF improvements with minimal performance penalty when there are enough cold tiles available and 2) use the DVFS technique when the thread migration cannot be used.

The input into the DRM algorithm includes temperatures of all the major modules of the tiles formed by cores (i.e., integer execution unit, caches, etc.) and NoC routers of the assumed regular mesh NoC as well as individual tile supply voltages. Temperatures and tile supply voltages are used by the neural network estimator to estimate lifetime reliabilities (as MTTF) of each tile containing a core and a router as well as of the overall CMP. Then, depending on the current number of cold tiles and on the comparison between the currently estimated MTTF with the desired target MTTF, the algorithm uses either the thread migration technique or the DVFS technique.

The logic behind the thread migration technique (see Fig. 4) is that if the currently estimated MTTF is less than the target MTTF, it moves threads from hot to cold cores to more uniformly balance the overall temperature profile, thereby increasing the current MTTF. In case that the estimated MTTF is higher than the target, no thread is migrated. The logic behind the DVFS technique (see Fig. 5) is that if the estimated current MTTF is less than the target MTTF, then, throttle the frequency of the core to the next lower frequency from the set of frequencies we work with (and lower its supply voltage too); otherwise, raise the frequency to the next higher frequency (and raise its supply voltage too); if the estimated current MTTF is within the vicinity (dictated though a user set parameter δ) of the target MTTF, then keep

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMSCS.2018.2870187, IEEE Transactions on Multi-Scale Computing Systems

Algorithm: DRM Controller

1: In: Desired $MTTF_{target}$; δ hysteresis bandwidth; γ maximum percentage of updated tiles in a control period; K repetition number
2: Out: Frequencies and supply voltages for all tiles and thread id running on each tile during next control period
3: Read in temperatures of all tiles of the CMP
4: Use neural network based MTTF estimator to calculate MTTF of each tile and of whole CMP
5: **if** $NumberColdTiles \geq NumberHotTiles$ **then**
6: **for** next K control periods **do**
7: Use thread migration technique
8: **end for**
9: **else**
10: **if** $0.8 \cdot MTTF_{target} < MTTF_{CMP} < 1.2 \cdot MTTF_{target}$ **then**
11: **for** next K control periods **do**
12: Use thread migration technique
13: **end for**
14: **else**
15: **for** next K control periods **do**
16: Use DVFS technique
17: **end for**
18: **end if**
19: **end if**

Fig. 3. Pseudocode of the proposed DRM algorithm. In our experiments, this algorithm is implemented as a callable routine inside the GEM5 simulation framework. Parameters δ, γ, and K can be set by the user to allow for calibration of how *aggressive* the DRM strategy is. The thread migration and DVFS techniques are described in Fig. 4 and Fig. 5. The values 0.8 and 1.2 were found empirically to provide good results.

Routine: Thread migration technique

1: **if** $MTTF_{CMP} < MTTF_{target} - \delta$ **then**
2: Sort all tiles in increasing order of their MTTF
3: **for** $i \leftarrow 1$ to $\gamma n/2$ **do** // n: number of tiles
4: **if** $MTTF_i < MTTF_{target} - \delta$ **then**
5: Migrate the thread in i^{th} tile to the $(n-i)^{th}$ tile and vice versa
6: **end if**
7: **end for**
8: **end if**

Fig. 4. Pseudocode of routine describing the thread migration technique called by the proposed DRM algorithm from Fig. 3

Routine: DVFS technique

1: **if** $MTTF_{CMP} < MTTF_{target} - \delta$ **then**
2: Sort all tiles in increasing order of their MTTF
3: **for** $i \leftarrow 1$ to γn **do** // n: number of tiles
4: **if** $MTTF_i < MTTF_{target} - \delta$ **then**
5: Switch down frequency and voltage of this tile
6: **end if**
7: **end for**
8: **else if** $MTTF_{CMP} > MTTF_{target} + \delta$ **then**
9: Sort all tiles in decreasing order of their MTTF
10: **for** $i \leftarrow 1$ to γn **do**
11: **if** $MTTF_i > MTTF_{target} + \delta$ **then**
12: Switch up frequency and voltage of this tile
13: **end if**
14: **end for**
15: **end if**

Fig. 5. Pseudocode of routine describing the DVFS technique called by the proposed DRM algorithm from Fig. 3

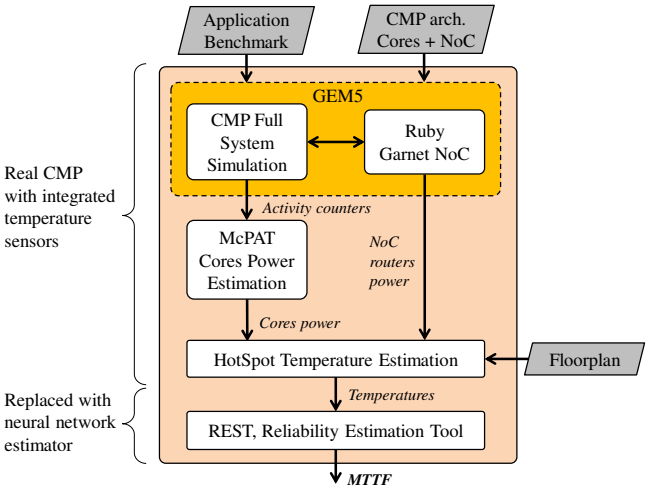


Fig. 6. Block diagram of the complete simulation framework to simulate a given application benchmark and to estimate lifetime reliability, measured as MTTF, of the entire system as combination of cores plus network-on-chip. Note that when the Rest tool is replaced by the neural network MTTF estimator, supply voltages are also provided together with temperatures as inputs to the estimator.

the same frequency for the core.

The time complexity of the algorithm from Fig. 3 is linear with the number of tiles, n, for which either thread migration or DVFS heuristic algorithms must be executed. Each of these requires a sort, that has $nlog(n)$ time complexity.

4 Simulation Results

4.1 Simulation Setup

Because we do not have access to NoC based CMP platforms with tens of cores that we investigate in this study (CMP architectures, which often times are exploratory), we resort like the rest of the research community to the next best way to test and validate our ideas, simulation tools. To test our DRM algorithm, we have developed our own simulation framework, which is implemented on top of the popular GEM5 full system simulation tool [28]. In addition, we integrate in our simulation framework several other point tools as shown in the block diagram from Fig. 6.

GEM5 tool is one of the most popular full system simulators capable of simulating entire computing systems constructed around singlecore or multicore processors. In the case of chip multiprocessors, the tool can model and simulate NoC communication between cores. It provides detailed timing and performance data and also integrates capabilities to estimate NoC router and link power consumptions. Hence, simulation of a given application benchmark is accurate and it also accounts for the operating system. Unfortunately, GEM5 tool does not output temperatures of cores, which in real processors would be available through temperature sensors integrated on chip. Therefore, we must use two additional tools in a sequence as shown in Fig. 6. Specifically, we first use McPAT power calculator [29] to compute power consumption values for cores based on the performance data (activity counters) from GEM5. Then, the power values of all cores and NoC routers are fed into the HotSpot temperature calculator [30] to estimate temperatures.

Finally, the temperature values are used as input into the Rest tool to estimate MTTF of each tile as well as of the entire CMP. The Rest tool is described in detail in our previous work [22]. We do not describe it here due to lack of space and because it is not a contribution of this paper, except

TABLE 1
Architectural configuration parameters.

Parameter	Value
Technology node	65nm
Frequencies	2GHz downto 1.5GHz, with 100MHz step
VDDs	1.1V downto 0.95V, with 25mV step
Core	Alpha EV6 21264
Core CPU model	Out of order (Detailed CPU)
Branch predictor	2 bit counter
Reorder buffer	80-entries
L1 ICache	32KB
L1 DCache	64KB
L2	2MB
Network	2D regular mesh, 1 router per core
Tile floorplan	Router to the top of core ALU occupies 20% of tile
Link bandwidth	32 bits
Routing algorithm	XY
Number of virtual channels (VCs)	2

to mention that it is based on a Monte Carlo simulation that works with failure times for TDDB and NBTI aging mechanisms modeled as Weibull distributions.

4.2 Simulation Parameters

In our simulations, we conduct experiments on several application benchmarks to investigate the proposed DRM algorithm for two different CMP architectures composed of 16 cores and 64 cores, respectively. Each of these architectures use 4×4 or 8×8 regular mesh networks-on-chip. The default architectural configuration parameters utilized in our custom GEM5 based simulations, unless otherwise specified, are shown in Table 1. The values of the parameters from Fig. 3 are $\delta = 10\%$, $\gamma = 50\%$, and $K = 4$. These values were found empirically to provide good results. Note that, aside temperature, the usage/duty cycle is another factor that can affect MTTF of a processor. For example, at an extreme, a CMP that is rarely used could last for many years and thus have a very large MTTF. Our work is under the assumption that the CMP architecture is exercised continuously, by the given benchmark being simulated. All the MTTF results reported in this paper are obtained under that assumption.

We would like to emphasize that within a simulation framework like this, we can perform any exploratory investigations for any chip multiprocessor architecture of interest. In addition, the simulation framework has the advantage of being able to stretch the ROI execution time for a given application benchmark in order to allow the complete sequence of processing steps illustrated in Fig. 6 to be performed. In real life deployment of the proposed DRM algorithm though, this sequence of steps would not be necessary because the on chip temperature sensors would provide temperature information directly. This is indicated on the left hand side of Fig. 6, where these processing steps would be shortcut by temperature sensor readings. The same figure shows that the Rest tool from this simulation setup would be replaced in real life deployment with a neural network based estimator described in detail, including training data generation and the training process, in our previous work [22].

Most of the previous DRM schemes did not consider the impact of the NoC component on the MTTF of the whole CMP system. This can result in errors in reliability estimation as high as 60%, as discussed in Section 2. Therefore, a direct comparison to such previous DRM schemes would not be an apple to apple comparison because they do not account for the effect of the NoC component. Other works such as [25] that also considered the whole system holistically focused on different CMP architectures (3D NoC based) and addressed different aging mechanisms such as electromigration. Hence, we present our results in comparison with the results we have achieved when no reliability optimization is done at all. In addition, we discuss the results in comparison with those that we obtained in our previous work with approaches that used only either thread migration or DVFS. In all our simulations, we consider both the computational (i.e., cores) and communication (i.e., network-on-chip) components in a unified manner for the purpose of the MTTF calculation.

4.3 Results

In this section, we report simulation results for several Parsec and Splash2x application benchmarks. In our simulations, we set as target or desired average MTTF a value that is with 100% longer than what it is when no DRM algorithm is used, which is our reference case. This target is in the range of 8-10 years for the studied testcases. In other words, we are interested in doubling the average lifetime of the investigated CMP architectures. As already mentioned, we investigate two different CMP architectures composed of 16 cores and 64 cores, respectively.

Fig. 7 shows the simulation results for *blackscholes* benchmark using 16 threads on a CMP architecture with 4×4 tiles. The plot shows only the period of time that covers the region of interest (ROI) of the GEM5 simulation. During each simulation, the GEM5 simulator is halted a number of times during the ROI (this number depends on the actual length of the ROI and the selected *control period*, which is the order of tens of ms or less in our case, due to the rather short duration of the ROI of all the investigated benchmarks) to perform DRM and to update the frequencies and voltages of each tile or to perform thread migration. Each of these stop-times corresponds to a data point out of the *sampling points* shown on the horizontal axis in Fig. 7. Note that in these figures the horizontal axis represents sampling points and not actual ROI execution time. That is because the actual length of the ROI portion when DVFS technique is used becomes longer in absolute time due to frequency throttling. This figure shows that the proposed DRM algorithm can bring and maintain the MTTF above or just beneath the desired objective (within δ parameter from Fig. 3). It can be noted that MTTF fluctuates around the target MTTF. That is because 1) of the variation in the workload that each core must do during different control periods inside the ROI and 2) of the inherent inertia when dealing with latent variables like temperature.

Fig. 8 shows the thermal profiles of the 4×4 CMP architecture in the reference case (i.e., no DRM algorithm being used) and in the case when the proposed DRM algorithm is used. These thermal maps correspond to the fifth sampling point from Fig. 7 and show that the DRM algorithm successfully manages to pull down the temperature of all tiles in the CMP architecture, thereby improving the MTTF of the entire system.

For an 8×8 CMP architecture, Fig. 9 shows the simulation results for *cholesky* benchmark compiled to use 64

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMSCS.2018.2870187, IEEE Transactions on Multi-Scale Computing Systems

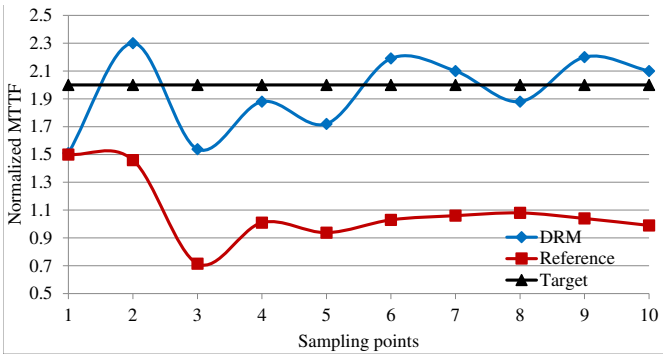


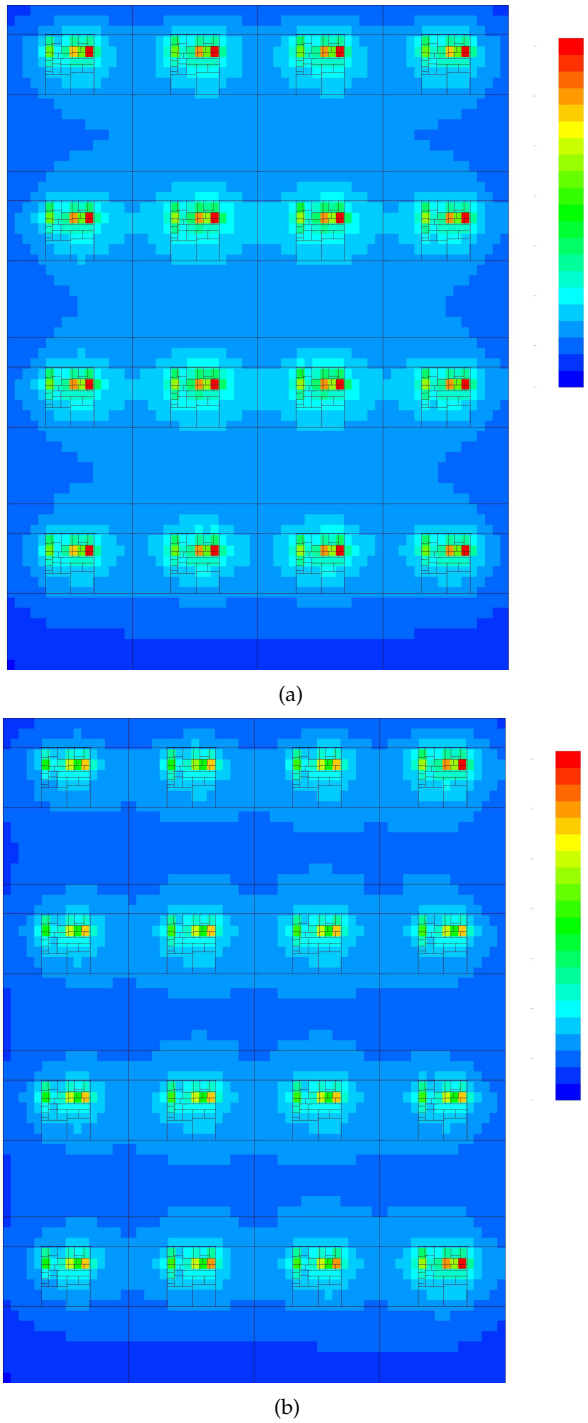
Fig. 7. Simulation results for *blackscholes* benchmark on an architecture with 4×4 tiles (i.e., 16 cores). Similar results were obtained for the other benchmarks.

threads. The dip formed by point 4-6s of the reference plot is because this particular benchmarks has a behavior that creates a workload pattern or traffic which is much heavier in the middle of the ROI period. During this dip, the activity counters registered inside the GEM5 simulations are much higher. This directly translates into increased temperature values that in turn trigger a degradation of the lifetime reliability. Likewise, Fig. 10 shows the thermal profiles, which again indicate that the DRM algorithm successfully manages to pull down the temperature of all tiles in the CMP architecture, thereby improving the MTTF of the entire system.

This is a good time when it is important to mention a subtle insight. The thermal maps from Fig. 8 or Fig. 10 are not the same as when we would do directly dynamic *thermal* management (DTM). That is because thermal management cannot be utilized directly as a proxy for lifetime reliability management. While temperature-wise a given core may be still within the acceptable limits, reliability-wise the core may be already in an emergency that requires immediate attention and vice versa. For example, the work in [8] showed that, when DVFS is utilized as the controlling technique, different frequencies are suggested by DRM and DTM schemes. The study found that at higher values of T_{qual} (qualifying temperature for DRM) and T_{max} (thermal design point), the frequency suggested by DTM would violate the system reliability requirement; while at lower values of T_{qual} and T_{max}, the frequency suggested by DRM would violate the system thermal requirement.

4.4 Discussion

Due to space limitations we cannot include here plots like Fig. 7 and Fig. 9 for all the benchmarks that we simulated. Instead, we present summary plots (see Fig. 11 and Fig. 12) that show the performance penalty and the change in energy delay area product (EDAP) for a user set target MTTF improvement of 100%, for each of the investigated benchmarks for both CMP architectures. For all simulated benchmarks, the target MTTF was reached. However, the achieved MTTF has fluctuations/oscillations around the target as observable in Fig. 7 and Fig. 9. In these plots, the reference for comparison is the case when no DRM algorithm is used. The proposed DRM algorithm successfully improves MTTF across the board while the performance



(a)

(b)

Fig. 8. (a) Thermal profile of the 4×4 CMP architecture running *blackscholes* benchmark with no DRM algorithm, (b) Thermal profile of the same architecture when the proposed DRM algorithm is used. The color-coded temperature range is $20°C$ (blue) to $120°C$ (red).

penalty is on average 7.7% and 8.7% for the two multi-processor architectures. While we do not have an exact breakdown of how much of the overhead is due to DVFS and how much is due to thread migration, we observed that DVFS has larger impact on the overall execution time. The EDAP values degrade a negligible amount, which is very desirable because from an EDAP stand point we lose very little while achieving significant improvements in lifetime reliability. The performance penalty is calculated based on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMSCS.2018.2870187, IEEE Transactions on Multi-Scale Computing Systems

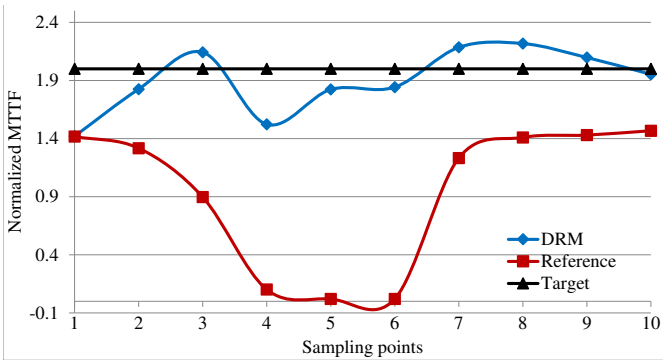
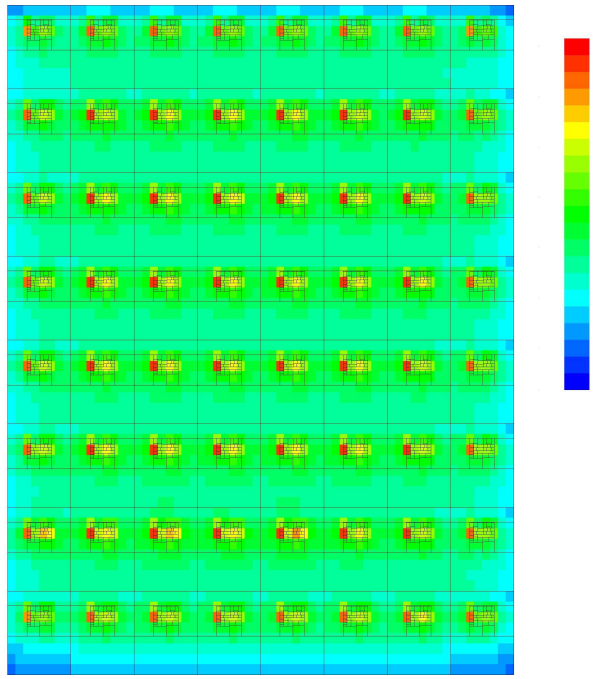


Fig. 9. Simulation results for *cholesky* benchmark on an architecture with 8×8 tiles (i.e., 64 cores). The MTTF of the reference case improves in the second part of the ROI because the actual workload decreases (some threads finish much earlier) for this particular benchmark.

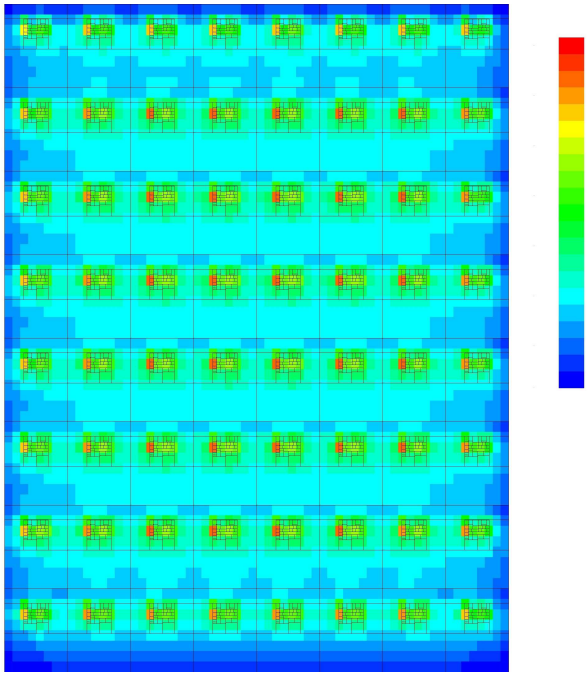
the difference in execution time inside the GEM5 simulator of a given benchmark. The GEM5 full system simulation tool reports the execution time, which usually becomes longer when we have enabled the proposed DRM technique. Despite the performance degradation on some benchmarks, our approach presents a method that combines both DVFS and thread migration and quantifies the tradeoff between performance and lifetime reliability; notably, it provides a means to tune this tradeoff to specific needs.

We notice that some applications are highly correlated in terms of their performance penalty and EDAP. This correlation is higher for example for benchmarks *lu-cp* and *fft* when executed on 8×8 CMP architectures. On the other hand, this correlation is higher for benchmarks *blackscholes* and *bodytrack* when executed on 4×4 CMP architectures. We suspect that this correlation could be in part due to the specific characteristics of traffic patterns that each benchmark creates on each core and through the network-on-chip. We noticed in our simulations that when the CMP system runs fully with all cores being busy all the time, it is very difficult to find room for improvement; no matter what one would do as DVFS or thread migration, the penalty in performance is more prevalent.

The proposed DRM algorithm performs better than each of the approaches when only either thread migration or DVFS would be employed. In our previous work from [22] we used only thread migration and the MTTF was improved by only up to 50%. The approach in [23] used only DVFS and MTTF was improved by up to 100% but with bigger performance penalties (up to 16%). The combination of both thread migration and DVFS techniques presented in this paper offers a better tradeoff between MTTF improvement and performance degradation.



(a)



(b)

Fig. 10. (a) Thermal profile of the 8×8 CMP architecture running *cholesky* benchmark with no DRM algorithm, (b) Thermal profile when the proposed DRM algorithm is used. The color-coded temperature range is $20°C$ (blue) to $120°C$ (red).

5 Conclusion

The proposed dynamic reliability management algorithm for chip multiprocessors combines thread migration and DVFS techniques to change the CMP operation such that the MTTF of the overall system is increased to the desired target with minimal performance degradation. Its straightforward software level implementation makes it cost effective and efficient. Full-system based simulations using a customized GEM5 simulator demonstrated that lifetime reliability can improved by 100% for an average performance penalty of 7.7% and 8.7% for 16 and 64 core NoC based chip multiprocessors compared to the case when no reliability optimization is done at all.

As future work, it would be interesting to study the use of more sophisticated reliability models in order to capture the fact that lifetime should also be a function of the current state of degradation [31].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMSCS.2018.2870187, IEEE Transactions on Multi-Scale Computing Systems

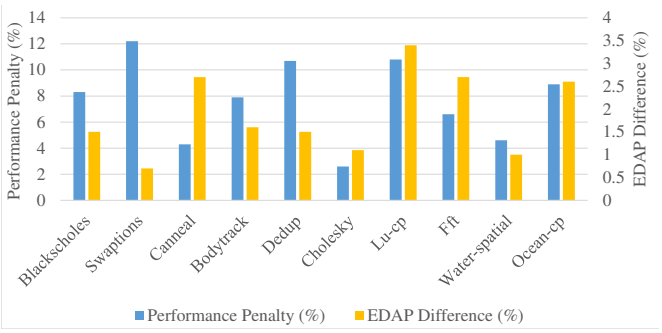


Fig. 11. Summary of simulations results for 4×4 CMP architecture for a target MTTF improvement of 100% (i.e., double lifetime). Each data point is the average of all values obtained during the hold times or sampling points illustrated in Fig. 7 for a given benchmark.

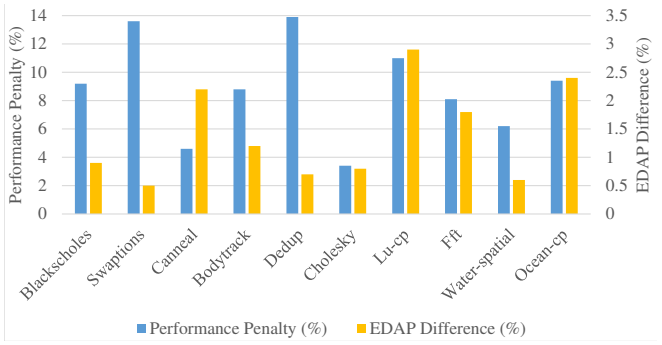


Fig. 12. Summary of simulations results for 8×8 CMP architecture for a target MTTF improvement of 100%.

ACKNOWLEDGMENT

This work was supported by NSF under grant CCF-1360439. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors.

REFERENCES

[1] The International Technology Roadmap for Semiconductors (ITRS), 2015 Edition, [Online]. Available: http://www.semiconductors.org/clientuploads/Research_Technology/ITRS/2015/5_2015%20ITRS%202.0_More%20Moore.pdf

[2] Failure Mechanisms and Models for Semiconductor Devices, JEDEC Publication JEP122E, 2009.

[3] B. Nikolic, Design in the power-limited scaling regime, IEEE Trans. on Electron Devices, vol. 55, no. 1, pp. 71-83, 2008.

[4] D. Park, C.A. Nicopoulos, J. Kim, N. Vijaykrishnan, and C.R. Das, Exploring fault-tolerant Network-on-Chip architectures, IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN), 2006.

[5] M.W. Rashid, E.J. Tan, M.C. Huang, and D.H. Albonesi, Power-efficient error tolerance in chip multiprocessors, IEEE Micro, pp. 60-70, 2005.

[6] A. DeHon, H.M. Quinn, and N.P. Carter, Vision for cross-layer optimization to address the dual challenges of energy and reliability, ACM/IEEE Design Automation and Test in Europe Conf. (DATE), March 2010.

[7] Z. Lu, J. Lach, M.R. Stan, and K. Skadron, "Improved thermal management with reliability banking," *IEEE Micro*, vol. 25, no. 6, pp. 40-49, Nov.-Dec. 2005.

[8] Jayanth Srinivasan, Lifetime reliability aware microprocessors, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 2006.

[9] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Multi-mechanism reliability modeling and management in dynamic systems," *IEEE Trans. on Very Large Scale Integration Systems (TVLSI)*, vol. 16, no. 4, pp. 476-487, April 2008.

[10] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, Maestro: orchestrating lifetime reliability in chip multiprocessors, Int. Conf. on High-Performance Embedded Architectures and Compilers (HiPEAC), 2010.

[11] A. Tiwari and J. Torrellas, Facelift: hiding and slowing down aging in multicores, ACM/IEEE Int. Symp. on Microarchitecture (MICRO), 2008.

[12] C. Zhuo, D. Sylvester, and D. Blaauw, Process variation and temperature-aware reliability management, ACM/IEEE Design Automation and Test in Europe Conf. (DATE), 2010.

[13] A.K. Coskun, R.D. Strong, D.M. Tullsen, and T.S. Rosing, Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors, SIGMETRICS/Performance, 2009.

[14] O. Khan and S. Kundu, A self-adaptive system architecture to address transistor aging, ACM/IEEE Design Automation and Test in Europe Conf. (DATE), 2009.

[15] P. Mercati, A. Bartolini, F. Paterna, T.S. Rosing, and L. Benini, Workload and user experience-aware dynamic reliability management in multicore processors, ACM/IEEE Int. Design Automation Conference (DAC), 2013.

[16] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli, Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems, ACM/IEEE Int. Design Automation Conference (DAC), 2014.

[17] H. Kim, S.B.K. Boga, A. Vitkovskiy, S. Hadjitheophanous, P.V. Gratz, V. Soteriou, and M.K. Michael, "Use it or lose it: proactive, deterministic longevity in future chip multiprocessors," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 4, Sep. 2015.

[18] M. Salehi et al., "dsReliM: Power-constrained reliability management in dark-silicon many-core chips under process variations," *CODES+ISSS*, 2015.

[19] M. Salehi et al., "Run-time adaptive power-aware reliability management for many-cores," *IEEE Design & Test*, 2017.

[20] D. Gnad et al., "Hayat: harnessing dark silicon and variability for aging deceleration and balancing," *DAC*, 2015.

[21] V. Rathore et al., "HiMap: A hierarchical mapping approach for enhancing lifetime reliability of dark silicon manycore systems," *DATE*, 2018.

[22] 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, Feb. 2014.

[23] M.G. Moghaddam, A. Yamamoto, and C. Ababei, "Investigation of DVFS based dynamic reliability management for chip multiprocessors," *IEEE Int. Workshop on Dependable Many-Core Computing (DMCC)*, 2015.

[24] V.Y. Raparti, N. Kapadia, and S. Pasricha, "CHARM: A checkpoint-based resource management framework for reliable multicore computing in the dark silicon era," *IEEE Int. Conference on Computer Design (ICCD)*, 2016.

[25] V.Y. Raparti et al., "ARTEMIS: an aging-aware runtime application mapping framework for 3D NoC-based chip multiprocessors," *IEEE Trans. Multi-Scale Computing Systems*, vol. 3, no. 2, pp. 72-85, 2017.

[26] A. Das, B.M. Al-Hashimi, and G.V. Merrett, "Adaptive and hierarchical runtime manager for energy-aware thermal management of embedded systems," *ACM Trans. Embedded Comput. Syst. (TECS)*, vol. 15, no. 2, pp. 1-25, 2016.

[27] H. Hong, J. Lim, H. Lim, and S. Kang, "Lifetime reliability enhancement of microprocessors: mitigating the impact of negative bias temperature instability," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, pp. 1-25, Sep. 2015.

[28] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewall, M. Shoaib, N. Vaish, M. D. Hill, and D.A. Wood, The GEM5 simulator, ACM SIGARCH Computer Architecture News Archive, 2011.

[29] S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi, McPAT: an integrated power, area, timing modeling framework for multicore and manycore architectures, IEEE/ACM Int. Symposium on Microarchitecture (MICRO), 2009.

[30] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron and M.R. Stan, "HotSpot: a compact thermal modeling method for CMOS VLSI systems," *IEEE Trans. on Very Large Scale Integration Systems (TVLSI)*, vol. 14, no. 5, pp. 501-513, May 2006.

[31] L. Huang and Q. Xu, "AgeSim: a simulation framework for evaluating the lifetime reliability of processor-based SoCs," *DATE*, 2010.