Microprocessors and Microsystems 38 (2014) 53–63

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Unified reliability estimation and management of NoC based chip multiprocessors

Alexandre Yasuo Yamamoto [a], Cristinel Ababei [b,*]

[a] Electrical and Computer Engineering Department, North Dakota State University, Fargo, ND 58105, USA
[b] Electrical Engineering, SUNY at Buffalo, Buffalo, NY 14260, USA

ARTICLE INFO

Article history:
Available online 23 November 2013

Keywords:
Reliability
Mean time to failure
Network-on-chip
Chip multiprocessor

ABSTRACT

We present a new architecture level unified reliability evaluation methodology for chip multiprocessors (CMPs). The proposed reliability estimation (REST) is based on a Monte Carlo algorithm. What distinguishes REST from the previous work is that both the computational and communication components are considered in a unified manner to compute the reliability of the CMP. We utilize REST tool to develop a new dynamic reliability management (DRM) scheme to address time-dependent dielectric breakdown and negative-bias temperature instability aging mechanisms in network-on-chip (NoC) based CMPs. Designed as a control loop, the proposed DRM scheme uses an effective neural network based reliability estimation module. The neural-network predictor is trained using the REST tool. We investigate how system's lifetime changes when the NoC as the communication unit of the CMP is considered or not during the reliability evaluation process and find that differences can be as high as 60%. Full-system based simulations using a customized GEM5 simulator show that reliability can be improved by up to 52% using the proposed DRM scheme in a best-effort scenario with 2–9% performance penalty (using a user set target lifetime of 7 years) over the case when no DRM is employed.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Due to continuous downscaling of CMOS technologies, several trends exacerbate the traditional design challenges in deep submicron domains. Increased aging mechanisms cause performance degradation and eventual device and system failure. Also, process variations increase the uncertainty of signal delays and result in variability of circuit performance and power. Moreover, increased device densities increase the circuit vulnerability to soft errors. Workload variations and dynamic power management techniques contribute to varying on-chip temperatures too. In addition, due to smaller supply voltages, the leakage power consumption increases and voltage noise margins decrease, hence affecting adversely reliability. These increasingly adverse factors lead to an increased number of transient, intermittent, and permanent errors. Aging mechanisms including time dependent dielectric breakdown (TDDB), negative bias temperature instability (NBTI), and electromigration (EM) are among the most increasingly adverse factors that can lead to delay errors and device breakdowns. Researchers from both industry and academia recognize that reliability is becoming a primary design concern in current integrated circuits [1,2]. Given that the most important factor through which these aging mechanisms affect chips is the temperature, 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.

In facing these increasingly adverse errors, we must address two main challenges: (1) Estimation of lifetime reliability and (2) Development of hardware and software techniques to improve system's lifetime.

Reliability has been addressed mainly by employing fault tolerance techniques that account for either the communication unit (i.e., bus or NoC) or the computation unit (i.e., cores). However, without considering both units in a unified approach, any previous reliability oriented design technique which focused on each unit separately (disregarding the presence of the other unit) is bound to be inaccurate, and thereby may lead to suboptimal designs. That is because, as it will be shown in the experimental results section, not treating the CMP as a whole, as the combination of two interacting and affecting each other units, significant errors can be introduced in the reliability estimations utilized during design optimizations. This is especially so in the case of the increasingly popular network-on-chip (NoC) that is utilized as the primary communication medium for CMPs with tens, hundreds, and even thousands of cores [3,4]. Because NoCs can occupy a significant portion of the overall CMP area, its impact on system reliability is significant.

* Corresponding author. Tel.: +1 716 645 1607; fax: +1 716 645 3656.
E-mail addresses: alexandre.yamamoto@ndsu.edu (A.Y. Yamamoto), cababei@buffalo.edu (C. Ababei).

0141-9331/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.micpro.2013.11.009

For a given set of resilience techniques, its effectiveness in achieving the desired system-level reliability must be evaluated, and its associated costs such as system-level energy and performance costs must be quantified. Currently, we are not aware of any previous work that evaluates system reliability in a unified manner, as a combination of both communication and computation units. Therefore, in this paper, we propose a new architecture level unified reliability evaluation methodology for CMPs. This methodology provides multiprocessor architecture designers with a framework that enables them to explore multiprocessor architecture characteristics and their impact on the mean time to failure (MTTF) as a measure of system's reliability. In addition, we develop a new *proactive* solution to improve system lifetime against TDDB and NBTI aging mechanisms. Our solution is based on a dynamic reliability management (DRM) scheme, which does thread migration as dictated by a neural network (NN) based predictor. The NN-based predictor is constructed and trained using the proposed reliability estimation methodology, which helps the proposed DRM scheme to effectively consider – in a unified manner – both the NoC and the cores as the main communication and computation units of CMPs. We want to emphasize that the focus of our work is CMPs whose lifetime reliability should be considered a problem in situations where they are utilized frequently (e.g., servers and database systems or increasingly prevalent portable devices from communications and gaming). In other scenarios (e.g., chips are utilized very rarely, possibly due to specific user behaviors) lifetime reliability is unlikely to be a problem because chips, in such cases, will remain healthy for very long times anyway.

The rest of this paper is organized as follows: In Section 2, we discuss related previous work and outline our main contributions. In Section 3, we present the models for TDDB and NBTI. In Section 4, we introduce the basic idea behind our proposed reliability estimation approach. A preliminary version of the reliability estimation framework appeared in a workshop paper [5]. In Section 5, we describe the proposed dynamic reliability management technique. Further details about the online estimation module of the DRM scheme are presented in Section 6. In Sections 7 and 8, we evaluate the proposed ideas and present our experiments. Finally, some conclusions are drawn in Section 9.

2. Related work and contribution

2.1. Previous work on reliability estimation

Significant work has been carried out to estimate the reliability of either single- and multi-processors [6–10] or of computer networks [11]. Reliability of NoCs has only recently been studied [12–14].

High-level metrics for reliable systems (e.g., reliability, availability, data integrity, mean time to failure (MTTF), mean time to repair (MTTR), architectural vulnerability factor (AVF), failures in time (FIT), FIT for reference circuit (FORC), etc.) have been used for quantifying the benefits of reliable systems. One popular metric, reliability function $R(t)$ denotes the probability that the system will operate correctly at time t. The MTTF, given by the relation $MTTF = \int_0^\infty R(t)dt$, has been one of the most popular measures for reliability. Increasing MTTF well beyond the expected useful life of a product is an important design objective.

The majority of previous lifetime reliability models assume a uniform device density on the chip and an identical vulnerability of devices to failure mechanisms. They also assume the lifetime distributions of failure mechanisms to be exponential [6–9], which is known to be inaccurate [15]. The assumption of exponential distributions only helps to simplify the problem of MTTF estimation, as it allows system level reliability to be calculated by applying the sum-of-failure-rates (SOFR) model [15]. The SOFR model is not realistic because failure rates increase with time due to aging. To address this issue, more general lifetime distributions can be utilized. For example, the authors of [6] utilize the lognormal distribution for their enhanced RAMP 2.0 tool. In this case, the analytical prediction of the system-level reliability becomes more difficult. Consequently, Monte Carlo simulations are typically employed [6,10].

Despite the significant work on modeling the lifetime reliability of computer networks and single- and multi-processors, there is no comprehensive methodology for assessing the reliability of NoC based multiprocessor SoCs. Designers should be able to answer questions about which units have the largest impact on system reliability and to validate that certain combinations of resilience techniques offer the optimal reliability for an application. The ability to perform such design activities depends on the availability of accurate and efficient metrics and tools. This is the main motivation for this paper.

2.2. Previous work on reliability management

The basic principle that underlies the vast majority of dynamic power [16–19] and thermal [20,21] management schemes is based on a control theory feedback loop. In this loop, certain variables that characterize the operation of the singlecore or the CMP system are continuously or periodically monitored. These variables are then utilized in specific optimization algorithms or decision logic to decide on particular measures that must be taken to drive the operation of the CMP system to the desired mode of operation. This principle is illustrated in Fig. 1, where previous work on dynamic management techniques is classified based on their implementation level into three main categories: software, hardware, and combined hardware/software.

The primary knobs shown in Fig. 1 include dynamic voltage and frequency scaling (DVFS), task/thread manipulation, and restructuring. All actions of these knobs translate in one way or another to reducing or migrating power dissipation while minimally degrading performance. From the perspective of what is utilized to trigger or guide management schemes, one can find commonly physical triggers like thermal or aging sensors and insights into thread information. Because power and temperature are well correlated with the workload behavior, significant effort was spent on workload predictors. In the context of dynamic power and thermal management, numerous approaches have been proposed to do real-time predictions. Notably, such examples include maximum likelihood estimation [16], recursive least square (RLS) [17], Bayes classifiers [18], autoregressive moving average (ARMA) [21], hypothesis testing [22], and neural networks (NN) [23–25].

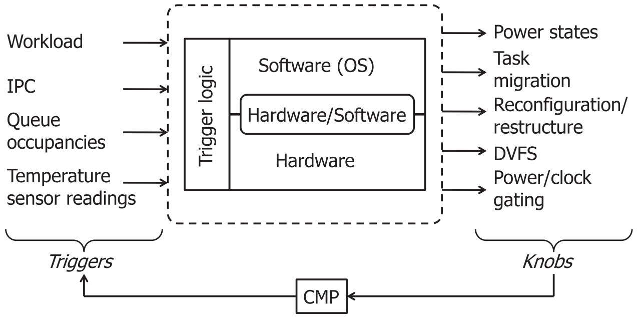

Fig. 1. Classification of dynamic management schemes into software, hardware, and hardware/software (HW/SW) approaches with illustration of commonly monitored variables and controlling knobs or mechanisms.

While significant previous work focused on power and thermal management, previous studies addressed reliability oriented design as well. The error-tolerance mechanisms, based on dynamic detection and correction of circuit timing errors, introduced by Razor [26] eliminate the need for voltage margins. The concept of dynamic reliability banking is proposed in [27] to address aging due to electromigration (EM). A concept related to [27], reliability slack, is leveraged by the DRM scheme in [28] to provide increased performance during periods of high processing demand. Based on readings from wearout sensors embedded in the CMP architecture, studies in [29] exploit the natural variation in workloads to assign jobs to cores in a manner that minimizes the impact of NBTI and TDDB on lifetime reliability. Facelift framework [30] addresses the problem of delay degradation due to wearout by hiding or slowing down the effects of aging. It hides aging through aging-driven application scheduling and slows it down by applying voltage changes at key times. A DVFS control and look-up table reliability estimation based DRM scheme is introduced in [31] for singlecore processors to address process variation aware oxide breakdown. The impact of job scheduling based power management on reliability is investigated in [32]. A dynamic tile partition algorithm is introduced in [33] to balance workload among active cores while relaxing stressed ones.

We identify several limitations of previous works. First, as discussed in the previous section, most of the previous works utilize reliability models [6–9] that assume the lifetime distributions of failure mechanisms to be exponential, which is not realistic. Many reliability estimation models are developed at device level and due to their complexity it is not clear how to apply them effectively at singlecore or multicore chip level especially dynamically. All previous reliability management schemes focus only on the computational portion of the system and neglect the communication component, which can represent a significant area of the overall system especially when increasingly popular NoCs are utilized. In addition, many previous works typically address only one aging mechanism. Finally, we are not aware of any approach that considers the CMP as a combination of the communication (buses or NoCs) and computation (cores) units.

2.3. Main contributions

We address the limitations discussed above by making the following contributions:

- We propose an architecture level unified reliability evaluation methodology for CMPs. At the core of the proposed reliability estimation engine lies a Monte Carlo algorithm which works with failure times for TDDB and NBTI modeled using Weibull distributions. While in this paper we focus on these two aging mechanisms only for simplicity, the proposed framework is general and it can be extended to include other aging mechanisms such as electromigration, stress migration, and thermal cycling.
- We integrate existing simulation tools to develop a full-system simulation framework and implement the proposed MC based reliability evaluation algorithm. We refer to the proposed Reliability ESTimation (REST) tool.
- We utilize REST to develop a new dynamic reliability management (DRM) scheme for CMPs. The proposed DRM scheme uses an effective neural network (NN) based reliability estimation module. This reliability predictor is trained using the REST tool.
- We explore the impact of NoC router layout on the MTTF of CMPs. We also investigate the system's MTTF when the NoC as the communication unit of the CMP is taken or not into consideration. We also demonstrate the proposed DRM scheme via full system simulations.

3. Lifetime failure models

As discussed in the previous sections, the assumption of exponential lifetime distributions for failure mechanisms is not realistic. To address this issue and to develop an accurate reliability model, more general lifetime distributions must be utilized. On the other hand, when using Weibull or lognormal distributions the analytical prediction of reliability becomes hard and therefore Monte Carlo simulations must be employed. In this paper, we adopt the Weibull distribution modeling for time dependent dielectric breakdown and negative bias temperature instability aging mechanisms, as these distributions have been found to best fit the corresponding physical wearout mechanisms [15].

3.1. Time dependent dielectric breakdown (TDDB)

Time dependent dielectric breakdown is caused by formation of a conducting path through the gate oxide to substrate due to electron tunneling current. TDDB has become increasingly severe as the thickness of the gate oxide decreased due to continuous technology downscaling. Under the same stress conditions, devices can feature directly hard breakdown or several soft breakdown events before the final hard breakdown [34]. While in this paper we employ the model studied in [6], the proposed reliability evaluation methodology is flexible and can be changed by replacing Eq. 1 with different models as they are discovered.

3.1.1. TDDB lifetime model

The model for $MTTF_{TDDB}$ is described by the following expression [6]:

$$MTTF_{TDDB} \propto \left(\frac{1}{V}\right)^{a-bT} \times e^{\frac{X+\frac{Y}{T}+ZT}{kT}} \tag{1}$$

where k is the Boltzmann's constant and a, b, X, Y, and Z are model fitting parameters and are determined from experimental data. In our implementation discussed later on, we use the same values as in [6] $a = 78$, $b = -0.081$, $X = 0.759\ \text{eV}$, $Y = -66.8\ \text{eV K}$, and $Z = -8.37\ \text{e}^{-4}\ \text{eV/K}$ based on the data from [35].

3.2. Negative bias temperature instability (NBTI)

Negative bias temperature instability mainly affects PFETs, when they are stressed at large negative gate voltages and high temperatures. NBTI manifests as a gradual increase in the threshold voltage and consequent decrease in drain current and transconductance [36]. The degradation exhibits logarithmic dependence on time. This effect has become more severe with technology downscaling, with the increase of the electric field applied to the gate oxide, and with the decrease of operating voltages.

3.2.1. NBTI lifetime model

The model for $MTTF_{NBTI}$ at a temperature T is described by the following expression [6]:

$$MTTF_{NBTI} \propto \left[\left(ln\left(\frac{A}{1+2e^{\frac{B}{kT}}}\right) - ln\left(\frac{A}{1+2e^{\frac{B}{kT}}} - C\right)\right) \times \frac{T}{e^{\frac{D}{kT}}}\right]^{\frac{1}{\beta}} \tag{2}$$

where A, B, C, D, and β are model fitting parameters. We use the same values as in [6] $A = 1.6328$, $B = 0.07377$, $C = 0.01$, $D = -0.06852$, and $\beta = 0.3$ based on the data from [37].

4. Proposed architecture level reliability evaluation methodology

4.1. Motivation

The key idea of the proposed time to failure evaluation methodology is to treat the CMP in a unified manner as a combination of communication and computation units. The motivation for this new approach is as follows. First, the area occupied by the NoC can represent up to 20% of the total chip area [38,39]. This is a significant portion of each tile (see Fig. 2) and can drastically impact power and temperature estimations. Second, the power consumption of the NoC can be as much as 25–40% of the overall chip power consumption [40,41]. The dissipation of this power can introduce hotspots that will affect the neighboring processing elements (PE) or cores and introduce errors in their temperature estimations. This problem is exacerbated when the PE of a tile is inactive (e.g., it is not processing any task), while its router is highly active due to the traffic between other source–destination communication pairs. For example, in Fig. 2, the processing element of tile $T10$ is affected by the traffic of $(t2, t7)$ and $(t3, t4)$ communication pairs, which contribute to the power consumption of the router $R10$. The unified model for reliability estimation proposed in this paper accounts for the behavior of the executing application and it will therefore capture the impact of workload variations on reliability.

4.2. Full system simulation framework

In order to implement and evaluate the proposed reliability evaluation methodology we construct a full-system simulation framework. The block diagram of the simulation framework illustrates the main steps of the proposed reliability evaluation methodology and is shown in Fig. 3. Its key components are as follows:

- As a multicore processor cycle-accurate simulator we utilize the GEM5 full-system simulator [42], which is a combination of M5 full-system simulator [43] and GEMS [44] (essentially Ruby with support for cache coherence protocols and interconnect models via Garnet [45]). GEM5 provides detailed timing and performance data and also integrates capabilities to estimate NoC router and link power consumptions. Therefore, simulation of a given benchmark is accurate as it accounts for the operating system as well.
- Performance data of each of the cores are then used as input to the power estimator McPAT [46]. The output of the McPAT power estimator is a list with power consumptions of each subblock of each core.
- Processors power consumptions provided by McPAT and the power consumption of individual routers of the NoC (provided by GEM5) are fed then to HotSpot [47]. HotSpot is an accurate and fast thermal model based on an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks. The output of the HotSpot simulation is a list of average temperatures for all NoC routers and for each subblock of all cores of the CMP.
- These temperatures are utilized together with the system level architecture floorplan by the Monte Carlo simulation engine to estimate the time to failure of the whole system. Details of this engine are presented in the next subsection.

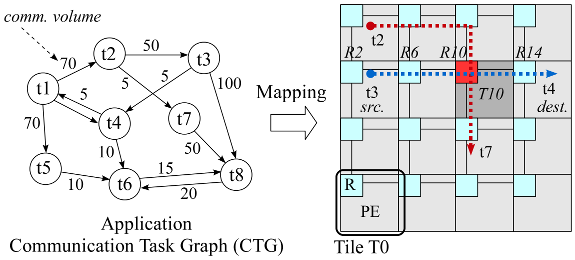

Fig. 2. Simplified diagram of an NoC based chip multiprocessor. Each tile is composed of a core or processing element (PE) and a router (R).

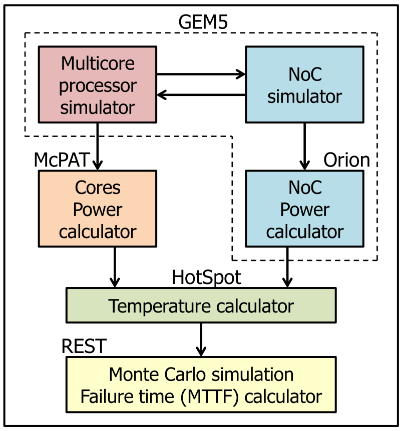

Fig. 3. Block diagram of REST tool framework: full-system simulation framework with integrated power consumption, temperature, and MTTF estimators.

4.3. Monte Carlo simulation based time to failure estimation

At the core of the proposed architecture level reliability evaluation methodology we employ a Monte Carlo (MC) simulation algorithm, which we implemented as a separate module in our simulation framework. The flow chart of the MC algorithm is shown in Fig. 4.

The input to the HotSpot temperature calculator is the floorplan of the CMP and power consumption of all subblocks: NoC routers and components of each processor core (e.g., ALU unit, L1 cache, etc.). We assume a regular tiled floorplan for the CMP and a regular 2D mesh NoC. The output of HotSpot is a list with temperatures for all routers and subblocks of each processor core. Note that these temperatures depend on the individual utilization of all cores

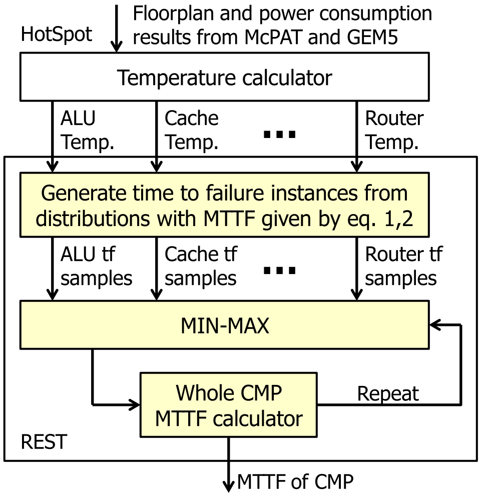
HotSpot
Floorplan and power consumption results from McPAT and GEM5
Temperature calculator
ALU Temp.
Cache Temp.
...
Router Temp.
Generate time to failure instances from distributions with MTTF given by eq. 1,2
ALU tf samples
Cache tf samples
...
Router tf samples
MIN-MAX
Whole CMP MTTF calculator
Repeat
REST
MTTF of CMP

Fig. 4. Block diagram of the Monte Carlo simulation based time to failure evaluation methodology for CMPs.

and routers as exercised by the application and its traffic. These temperatures are plugged into Eqs. (1) and (2) from Section III. These equations model the mean time to failures of the probability distributions associated with each router and core subblock, from which we draw samples (or instances) during the Monte Carlo iterations.

The MC algorithm (see also Fig. 5) proceeds with the following main steps (1) for each failure mechanism run $N = 10^5$ simulations: (a) for each subblock, generate failure time instances from the corresponding distribution and (b) use MIN–MAX analysis of these times according to the system's configuration to calculate the time to failure tf^j_{min} during simulation iteration $j = 1, \ldots, N$. (2) Calculate the time to failure for the current failure mechanism as $tf_l = (\sum_{j=1}^{N} tf^j_{min})/N$. (3) Calculate the value of the overall MTTF or time to failure of the CMP as the minimum among the failure times due to each failure mechanism. We selected $N = 10^5$ because in our experiments we found that this number is a good tradeoff between computational runtime and statistical significance of results.

4.4. Generation of samples from a Weibull distribution

During each MC simulation iteration, we need to generate random *instances* of failure times for each subblock. This is realized by the *generate_instance*($MTTF_l$) procedure called in line number 8 in Fig. 5, which draws samples from Weibull distributions whose means are given by Eqs. (1) and (2). Because the Weibull cumulative distribution function is given by:

$$F(x) = 1 - e^{-(\frac{x}{\alpha})^{\beta}} \quad (3)$$

one can generate samples via the expression:

$$x_{sample} = \alpha \cdot [-ln(1 - u)]^{\frac{1}{\beta}} \quad (4)$$

where $u = rand(0,1)$ is a random number generated uniformly from the interval $[0,1]$. α and β are the scale and the shape factors characterizing the Weibull distribution. In our implementation of *generate_instance*($MTTF_l$), we utilize a value of $\beta = 1.64$ as in [48] while *alpha* is derived from the expression of the mean of a Weibull distribution:

$$\alpha = \frac{MTTF_l}{\Gamma(1 + \frac{1}{\beta})} \quad (5)$$

where $\Gamma(\cdot)$ is the Gamma function.

```
Algorithm: Monte Carlo algorithm
1: In: CMP floorplan and power consumption of all subblocks
2: Out: Estimate of MTTF of whole CMP
3: for $l \leftarrow 1$ to $F$ do // F: number of failure types
4:     Calculate $MTTF_l$ using equations from Section III
5:     for $j \leftarrow 1$ to $N$ do // $N = 10^5$ Monte Carlo iterations
6:         $tf^j_{min} \leftarrow INF$ // Initialize
7:         for $k \leftarrow 1$ to $S$ do // S: number of subblocks
8:             $tf_k \leftarrow generate\_instance(MTTF_l)$
9:             if $tf_k < tf^j_{min}$ then // Generalization: MIN_MAX
10:                $tf^j_{min} = tf_k$
11:            end if
12:        end for
13:    end for
14:    $tf_l = \frac{\sum_j^N tf^j_{min}}{N}$
15: end for
16: return $tf = MIN\{tf_l\}$ // Estimate of MTTF of whole CMP
```

Fig. 5. Algorithm pseudocode of the Monte Carlo simulation.

5. Proposed dynamic reliability management scheme

In this section, we introduce the proposed dynamic reliability management (DRM) scheme, which we develop using the reliability estimation technique discussed in the previous section. The block diagram of the proposed DRM scheme is shown in Fig. 6. We introduce some notations first. We assume that the whole control loop from Fig. 6 is executed periodically with a control period (also referred to as a design epoch) of T and that the CMP has N tiles. $t_i(k)$ is the average temperature of tile i in the k^{th} control period. $\Delta t_i(k)$ is the difference between the temperatures in current $t_i(k)$ and previous $t_i(k-1)$ control periods, i.e., $\Delta t_i(k) = t_i(k) - t_i(k-1)$. It represents the recent temperature-trend history of tile i. The reliability of an individual tile i is $r_i(k)$ while $r(k)$ is the reliability of the whole chip. Reliability is measured as the mean time to failure (MTTF). The user specified target (i.e., set point) reliability is R_s. The goal of the control loop in Fig. 6 is to guarantee that $r(k)$ converges to R_s within a given settling time.

The two main components of the proposed DRM scheme are the reliability estimation module and the migration controller. The reliability estimation module provides online estimation of reliability, which is compared with the user target reliability. Based on this comparison, then, the migration controller identifies threads to be migrated such that the CMP reliability is driven towards the target reliability. These two components are discussed in the following sections.

5.1. Reliability estimation module

The difficulty in designing the reliability estimation module stems from the fact that lifetime reliability is the result of intricate relations between the temperature profile, power dissipation, system architecture, workloads, traffic, process and supply voltage variations, uncertainty of signal delays, etc. Nevertheless, it is intuitive to observe that, under the assumption for example of using thread migration as the controlling knob in Fig. 1, reliability can be improved by relocating threads scheduled to run on hot cores to other cores whose temperature is currently lower. This is similar to previous studies, which did that for controlling the temperature profile. For example, in the simplified representation of a 16 cores chip multiprocessor from Fig. 6, threads scheduled on the hottest core can be migrated (as indicated by the arrows) to cores with lower temperatures. What adds to the difficulty of reliability estimation is that thermal management cannot be utilized directly to also do reliability optimization. For example, 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. Srinivasan [6] showed that, when DVFS is utilized as the controlling knob, different frequencies

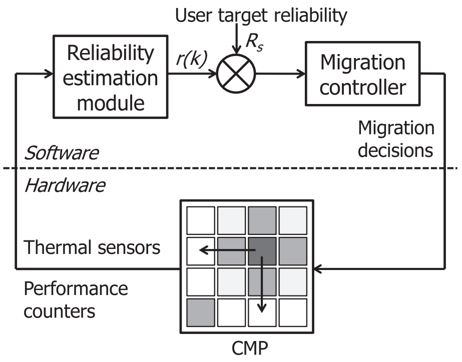

Fig. 6. Control loop of the proposed dynamic reliability management scheme. Threads executing or scheduled to be executed on core at hotspot location can be migrated to cooler cores to alleviate temperature, thereby reliability.

are suggested by DRM and dynamic thermal management (DTM). They 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.

To address the above difficulties, we build our reliability estimation module using a neural network (NN) based approach. Our motivation is based on the fact that the NN approach applied to the case of hardware-based thermal simulation can provide short computational delays (for example, in the order of a few gate delays), low resource usage, and low error margins [23]. We find that this approach can be applied to reliability estimation as well provided that the appropriate training is done and estimation is done for relatively short horizon spans. We want to emphasize that while NN-based models have been used in the context of thermal management, we are not aware of any previous work that used NN models for reliability management. Aside from being accurate in our estimations, we want to achieve fast response times to target decision epochs in the order of seconds or less in order to be able to react faster and more effectively to workload variations. Note that this is not necessary when the decision epoch is selected intentionally long (in the order of weeks in [49]). In such cases, one can afford to employ directly simulation-based reliability estimation techniques as the estimation module. We implement the NN model in software to minimize hardware changes and to be able to deploy it on existing hardware systems. A more detailed description of the proposed reliability estimation module is presented Section 6.

5.2. Migration controller

The primary role of the migration controller from Fig. 6 is to decide when threads should be migrated and to what cores, how many threads, how far from the initially scheduled cores, and if done in a cascading fashion or not to limit introducing large delay penalties. In our implementation, the period of updates varies as function of the application being run (updates are triggered by situations when cores' MTTF is below the desired target) but it is set to be always longer than the lower bound of 30 ms in order to achieve fast convergence to the desired target reliability while keeping performance penalty acceptable. We implement the migration controller described in Fig. 7 for its simplicity, which in turn results in fast response times. Migration decisions (illustrated in Fig. 8) are actuated periodically at every temperature sampling interval. These decisions are made based on the priority of all scheduled threads. The priority of threads is given by their rank in the sorted list. Migration is done while the current reliability has not reached the target reliability. However, there is no guarantee that the target reliability will be reached, especially when the target is set too aggressively by the user (we will show an example in the results section). Therefore, our approach to regulate lifetime reliability is a *best-effort* approach, meaning that the DRM scheme will migrate threads to track the target reliability only if it is worth to do so. For example, it is possible that all cores are already hot and migration alone would not help. In such cases, extensions based on DVFS or simply stalling the system can address such situations.

```
Algorithm: Migration controller
In: Difference between estimated and target reliabilities;
    tile temperatures and reliabilities
Out: Migration decisions
Sort all tiles in increasing order of their individual MTTF
Migrate threads from cores in tiles whose MTTF is below
    the target to closest cooler cores
```

Fig. 7. Pseudocode of the algorithm that implements the migration controller from Fig. 6.

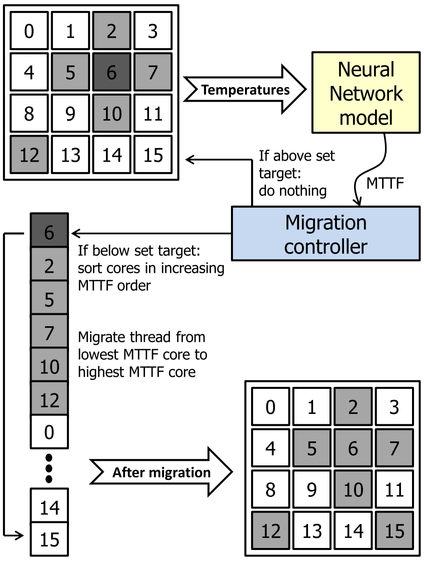
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Temperatures
Neural Network model
If above set target: do nothing
MTTF
Migration controller
If below set target: sort cores in increasing MTTF order
6
2
5
7
10
12
0
14
15
Migrate thread from lowest MTTF core to highest MTTF core
After migration
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Fig. 8. Illustration of thread migration from heated core 6 (lowest MTTF) to core 15 (highest MTTF). Target reliability is a user set parameter; set based on the desired tradeoff between achievable MTTF and performance penalty.

6. Online reliability estimation module

In this section we discuss in more details the reliability estimation module introduced in the previous section.

6.1. Neural network based reliability estimation

Neural network (NN) models have been utilized in numerous applications (e.g., pattern recognition and data classification) due to their ability to extract patterns and detect trends in complex or imprecise data. A neural network is composed of a number of interconnected neurons (processing elements) working together to solve a specific problem. First, it must be trained through a standard learning process. Then, the NN model can be utilized to provide estimations on new data of interest. The proposed online reliability estimation module is constructed using a neural network model, using an approach similar to [24,25]. Note that the studies in [24,25] utilize a NN in a different context, of thermal management. Here we apply it to reliability management, where it effectively mimics compactly the system from its lifetime reliability perspective. The neural network can utilize as inputs the current temperatures, supply voltages, and clock frequencies of each tile as well as their variation trends (i.e., recent history). The output of the estimation module consists of the values of the reliability of each tile as well as the reliability, r_k, of the whole CMP in the kth control period.

Generally, a neural network model may have several layers. Each layer implements the transfer function $\mathbf{Y} = f_i(\mathbf{WX} + \mathbf{b})$, where $\mathbf{W}$ is a weight matrix, $\mathbf{b}$ is a bias vector, and $\mathbf{X}$ and $\mathbf{Y}$ are the input and output vectors of layer i. The size of $\mathbf{W}$ is $m \times n$, where n is the number of inputs and m is the number of neurons in this layer. The size of $\mathbf{b}$ is $m \times 1$. In our case, we use a two-layer neural network whose block diagram is shown in Fig. 9. We use *tansig* and *purelin* as the transfer functions $f_1(\cdot)$ and $f_2(\cdot)$ in Fig. 9. The output of these transfer functions are given by the expressions:

$$y = \frac{2}{1 + e^{-2\left(\sum_{j=1}^{2N} w_j \cdot x_j + b_1\right)}} - 1 \tag{6}$$

$$r(k) = y \cdot w_2 + b_2 \tag{7}$$

The relevant features that represent the inputs x_j to the neural network are the temperatures $t_i(k)$ and the supply voltages $VDD(k), i = 1, \ldots, N$ of all tiles. In our current implementation the supply voltages of all cores remain constant; they are included in the model for future enhancements based on DVFS.

6.2. Training the neural network

A crucial aspect of the neural network based estimator is the training to compute the weight terms. Learning in NNs consists of finding the correct set of weights, such that the input–output relationship of the system is emulated accurately for all possible cases. The training process is done at design-time using a set of representative workloads or via sweeping input variables within ranges of interest. To do training, we employ the REST reliability estimation technique discussed in Section 4. In our case, training is performed by supplying to the NN a finite set of inputs and outputs from the REST framework over a specific simulation interval (N_{train} time points) as shown in Fig. 10. During each iteration, the output from the neural network model and the target output from REST are compared and based on their difference (i.e., error), the weights of the NN are updated using the training algorithm. In our implementation, we use the neural network fitting tool *nftool* of Matlab for the training algorithm.

Using the REST full system simulator and reliability estimation framework, we generate N_{train} inputs for the NN model (shown as $INPUTS(t_{k+1})$ in Fig. 10). For each of these inputs we also compute the reliability $r_{REST}(t_{k+1})$ using the Monte Carlo based engine of the REST framework. In this way, we effectively generate reliability traces.

It is important to note that because training is done using the REST tool at design-time, the NN-based reliability estimator inherently accounts for the contribution to reliability of both the communication unit (i.e., network-on-chip) and the computational units (i.e., cores). This is a unique advantage of our DRM scheme in contrast with previous works, which focus only on either cores or NoC. While in our current implementation of the proposed DRM scheme the NN is trained at the design time only, we see no reason for which one could not retrain the NN model periodically (e.g., monthly) to help improve accuracy, provided that new training data becomes available.

7. Results – REST tool

In the first set of experiments, we demonstrate the proposed reliability evaluation methodology on Parsec benchmarks [50]. The architectural configuration parameters utilized in our simulations are shown in Table 1. All other parameters are kept at their default values inside the GEM5 full-system simulator.

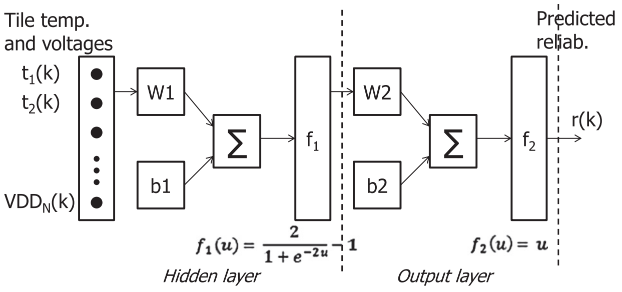

Fig. 9. Architecture of neural network model.

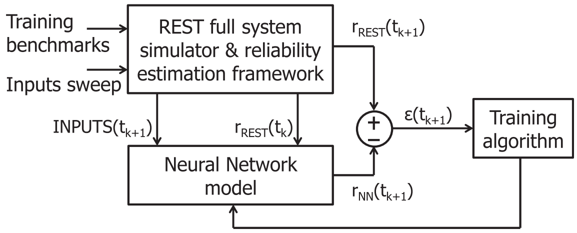

Fig. 10. Illustration of the training process of the neural network model using REST reliability estimation framework.

7.1. Router location within the tile

In this set of experiments, we investigate the impact of NoC router location within the floorplan of a single tile on the MTTF of the overall CMP. We consider two simple tile layouts as shown in Fig. 11. While the area occupied by a router depends primarily on the buffers size and the number of ports, based on the discussions and designs in [38,39], we assume a router whose area is 20% of the area occupied by the processor core within a tile.

The comparison between the MTTFs achieved in these two different cases for each of the simulated benchmarks on multicore architectures with 4, 16, and 64 cores is shown in Fig. 12. We observe that when the router is located in the upper part of the tile, as shown in Fig. 11.a, the system's MTTF is smaller. Because in the case shown in Fig. 11.b the router is further away from the actual Alpha core, the thermal profile of the overall system is better. However, the difference is rather small; we suspect as the main reason the benchmarks, which do not create a lot of traffic through the network. Nevertheless, when the router is located in the upper part of the tile closer to the Alpha core, it still represents a poorer heat sink (due to its own higher temperature) for the heat diffused from the core.

7.2. Network impact

Here, we investigate the impact of taking into consideration the NoC (as the communication unit of the CMP) on the MTTF of the overall CMP. In other words, we want to see with how much is the MTTF optimistically estimated by previous reliability models, which did not consider the network. The comparison between the MTTFs achieved in these two different cases for each of the simulated benchmarks on multicore architectures with 4, 16, and 64 cores is shown in Fig. 13. In both cases we utilize the tile layout from Fig. 11b.

Table 1
Architectural configuration parameters.

Parameter	Value
Core (frequency, VDD)	Alpha EV6 21264 (1 GHz, 2 V)
Branch predictor	2 Bit counter
Reorder buffer	80-entries
L1 ICache	32 KB
L1 DCache	64 KB
L2	2 MB
Network	2D regular mesh, 1 router per core
Link bandwidth	32 bits
Routing algorithm	XY
Number of virtual channels (VCs)	2

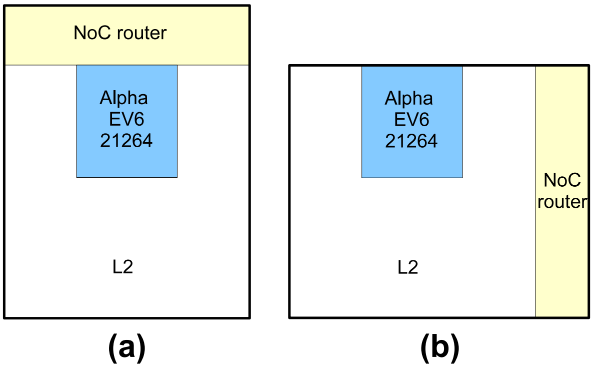

Fig. 11. Tile layouts with different locations for the NoC router: (a) top router, (b) side router.

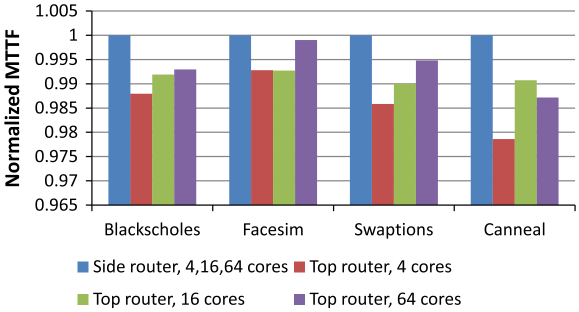

Fig. 12. Relative comparison of the CMP's MTTF achieved for two different locations of the NoC routers within a tile. The two different NoC routers are shown in Fig. 11.

We observe that when the NoC is not taken into account during the lifetime evaluation process, the MTTF of the overall CMP is with up to 60% longer than when the network is included. This is not surprising, as previous work found that networks and processors alone can reach peak temperatures of 68.6 °C and 77.9 °C, respectively, while when networks and processors are jointly considered, chip peak temperature can reach 104.7 °C [51].

7.3. Discussion

These results demonstrate the importance of considering the combination of both computation and communication when evaluating reliability of CMPs. If this is not done, then the errors became unacceptably large and any optimization strategy may be significantly off from the real desired targets. In addition, these results confirm that it is beneficial form a reliability perspective to floorplan the main components of a CMP and its cores such that hot spots are uniformly distributed across the entire area of the chip.

Because GEM5 is a sophisticated and capable simulation platform (it can simulate the Linux operating system as well as a variety of core types and NoC topologies), the proposed reliability evaluation framework can be utilized to explore a large variety of design tradeoffs and techniques spanning multiple layers. For example, designers could investigate dynamic voltage and frequency scaling, activity throttling, workload migration/scheduling among cores, and network traffic migration via adaptive routing as mechanisms or knobs to control and regulate the power/thermal profiles of the overall chip. Along these lines, in the next section, we report results achieved with the proposed dynamic reliability management scheme.

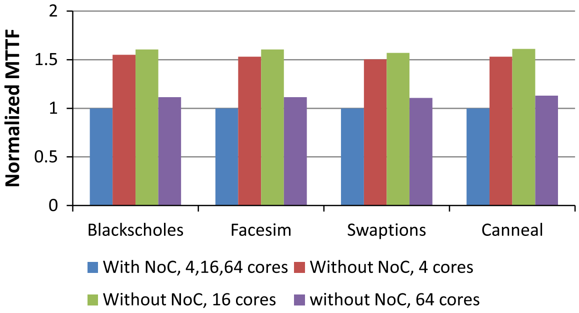

Fig. 13. Relative comparison of the CMP's MTTF achieved when the network is taken or not into consideration during the reliability evaluation process. Results are obtained for the side router layout from Fig. 11b.

8. Results – DRM scheme

In the second set of experiments, we demonstrate the proposed dynamic reliability (DRM) scheme. We do not compare our results to previously studied DRM schemes because previous studies did not consider the effect of the NoC, which can result in differences in reliability estimation as high as 60%, as shown in the previous section. This could result in the comparison of design points from completely different locations in the design space (with the design point obtained using previous solutions being evaluated with large errors due to the 60% estimation differences). In addition, none of the previous DRM schemes is publicly available. We want to emphasize that if we did not consider the contribution of the NoC to the system's reliability either, we would get reliability estimations that would be similar to previous works, which also utilized Monte Carlo simulation based techniques [6]. Therefore, we avoid replicating previous results here. In our experiments, we are interested in getting insights into the amount of achievable lifetime reliability improvement over the case when no DRM scheme is employed at all. We utilize the proposed DRM scheme on the same benchmarks as in the previous section. The experiments are performed using a modified version of the GEM5 full system simulator. We modified the GEM5 simulator to integrate the proposed DRM scheme and to be able to collect reliability traces. We generate offline the neural network based reliability estimation module, plug it into the DRM scheme, which is then integrated with the GEM5 simulator for dynamic usage during runtime. Our investigations follow the experimental setup described in Fig. 14. In all our simulations, we utilize a 16 cores CMP organized as a 4 × 4 array of tiles interconnected via a mesh network-on-chip. The configuration details of the simulated CMP are the same as in the previous section, shown in Table 1. Because

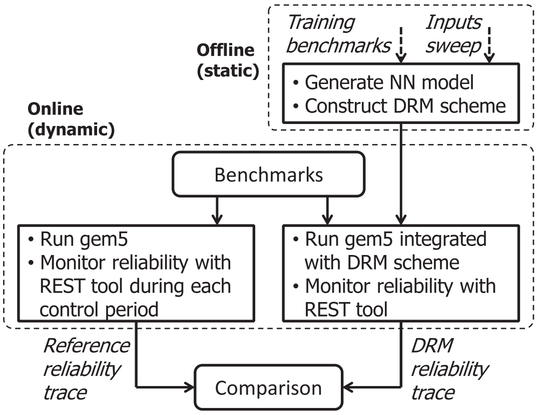
Training benchmarks
Inputs sweep
Offline (static)
• Generate NN model
• Construct DRM scheme
Online (dynamic)
Benchmarks
• Run gem5
• Monitor reliability with REST tool during each control period
• Run gem5 integrated with DRM scheme
• Monitor reliability with REST tool
Reference reliability trace
DRM reliability trace
Comparison

Fig. 14. Flow diagram of the experimental setup.

we use XY routing, the NoC is guaranteed not to face deadlock situations even after thread migrations.

Fig. 15, Fig. 16, and Fig. 17 show the simulation results for *swaptions*, *blackscholes*, and *facesim* benchmarks run as applications with 16 threads. In all our simulations, unless otherwise stated, the user set lifetime reliability target is 7 years, which is similar to those utilized in related literature. Similar plots are obtained for other Parsec benchmarks. The plots show only the period of time that covers the region of interest (ROI) of the GEM5 simulation. Table 2 summarizes the information presented in these plots. The performance penalty includes also the cost for executing the NN based predictor, whose contribution is less than a tenth of this overall penalty.

These figures show that the proposed DRM scheme can bring and maintain the reliability above or just beneath the target reliability. However, the achieved reliability when DRM is turned on is not always above the desired target. This is because the DRM scheme cannot bring the system to a state where the target reliability is met if the target is set unrealistically too high. For example, we set a higher target reliability for the *canneal* benchmark (7 years instead of five), whose simulation result is shown in Fig. 18. In this case, the DRM scheme does its best in its effort to reach the target reliability and consistently maintains MTTF with an average of 52.2% above the reference run. The reason for not reaching the target reliability is that we set a bound constraint on the acceptable performance penalty. In cases where performance is not important, threads can also be delayed (not only migrated) to allow cores to cool off and drive the reliability up in this way. One can simply delay threads' execution considerably to achieve very high reliabilities but at the cost of severe performance penalties; at the limit, one can stop all threads indefinitely and achieve infinite MTTF. However, we believe that an effective way to address this issue is to also use DVFS. Lowering voltages and frequencies would result into smaller performance penalties than excluding cores and would improve the chance to reach reliability targets. This investigation is left to future work.

8.1. Discussion

We notice that the performance penalty can sometimes become large (e.g., 9%). This is due primarily to the increased thread migrations dictated by the migration controller. This performance penalty can be lowered by setting larger control periods for the DRM scheme and by limiting the number of migrations. However, this slows down the response time in tracking the target reliability. The DRM scheme provides a mechanism to trade-off performance for improved lifetime reliability; as it is challenging to achieve longer lifetimes without paying any price. That is because the DRM scheme adds extra costs required to implement it, which may be in software, hardware, or both. One could, however, work within the context of performance where *user satisfaction* is also taken into consideration. For example, recent work has included user satisfaction as a design concern too [52,53]. In such contexts, one can opportunistically reduce energy consumption at the expense of actual performance decrease but without affecting user satisfaction. In this case the user perceived performance is virtually as good as when nothing was done to save energy. These ideas could

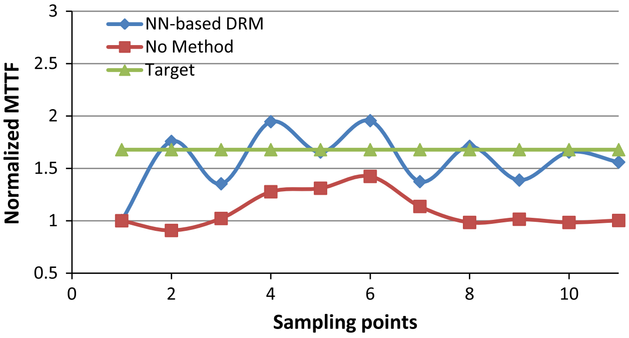

Fig. 15. GEM5 simulation of *swaptions* benchmark.

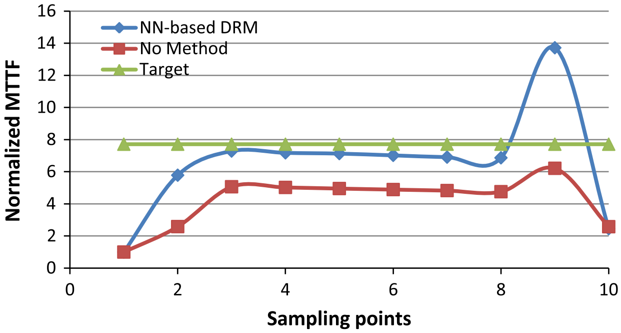

Fig. 16. GEM5 simulation of *blackscholes* benchmark.

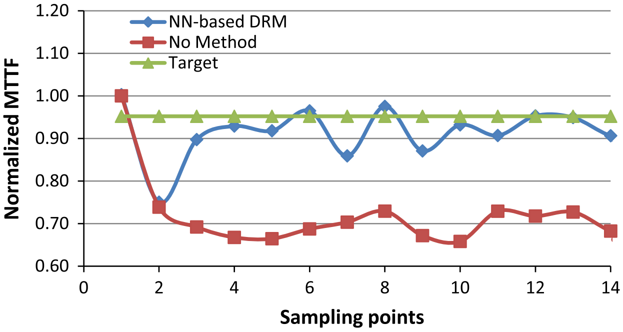

Fig. 17. GEM5 simulation of *facesim* benchmark.

Table 2
Summary of simulations shown in Fig. 15, Fig. 16, Fig. 17, and Fig. 18.

Benchmark	MTTF improv. (%)	Perf. Penalty (%)	ROI exec. time (baseline run) (ms)	ROI exec. time (DRM run) (ms)	GEM5 sim. time (h)
Swaptions	45.24	2.94	781	804	7
Blackscholes	50.12	6.51	568	605	6
Facesim	32.7	7.28	2319	2448	24
Canneal	52.2	9.16	480	524	12

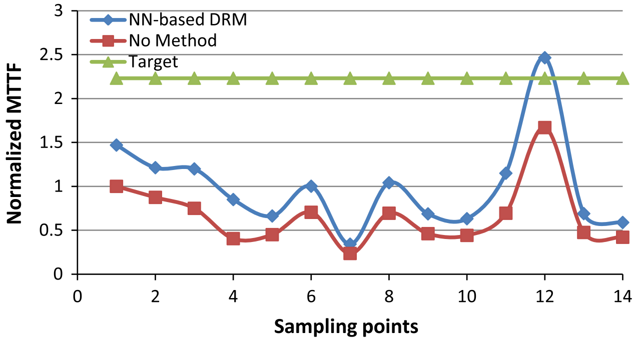

Fig. 18. GEM5 simulation of *canneal* benchmark.

be extrapolated to the case of reliability (instead of energy) optimization as well in order to virtually not degrade the perceived performance; however, this is outside the scope of this paper.

We would like to emphasize that the computational complexity of the proposed DRM scheme is very low (in the order of a few ms) and given by the times required by the NN-based estimation module and the by the thread migration controller, which at its turn is dominated by a simple sorting algorithm. The computational runtime bottleneck is the static simulation time required at design time, when the GEM5 simulator requires several hours or more to simulate a benchmark. However, this is done only at design time and possibly periodically at long time intervals (e.g., months). More importantly, as already mentioned, GEM5 represents the state of the art in full system multicore simulation. It represents one of the main drivers in processor research as it allows pre-silicon architectural explorations. Therefore, dynamic reliability schemes like the one presented in this paper can be investigated for processor architecture before they are actually implemented in silicon.

9. Conclusion

The two main contributions of this paper are: (1) an architecture level unified reliability estimation methodology for CMPs and (2) a dynamic reliability management (DRM) scheme for network-on-chip based chip multiprocessors. The proposed reliability estimation, REST, tool is motivated by the fact that each of the communication and computational units of multicore processors may become a reliability bottleneck. At the core of the reliability estimation engine lies a Monte Carlo algorithm. We investigated how system's lifetime changes when the NoC as the communication unit of the CMP is considered or not during the reliability evaluation process and found that differences can be as high as 60%. Furthermore, we developed a new dynamic reliability management (DRM) scheme to effectively control lifetime reliability of network-on-chip based CMPs and address TDDB and NBTI aging failure mechanisms. Designed as a control loop, the proposed DRM scheme uses an effective neural network based reliability estimation module, whose training is done with the proposed REST tool. Hence, one of its main merits is that the online reliability estimation is done in a unified manner considering both cores and network. Simulation results showed that reliability can be improved by 50% in a best-effort scenario with 2–9% performance penalty.

As future work, we want to also use DVFS in combination with thread migration and to employ techniques for full-system simulation speed-up. Another interesting idea is to consider also the hard or soft deadlines for the application execution aside from just reliability estimation. Thus, for applications with hard deadlines, a better way would be to study and use DVFS considering task deadlines and reliability and then perform task or thread migration.

References

[1] S. Borkar, Designing reliable systems from unreliable components: the challenges of transistor variability and degradation, IEEE Micro 25 (6) (2005) 10–16.
[2] A. DeHon, H.M. Quinn, N.P. Carter, Vision for cross-layer optimization to address the dual challenges of energy and reliability, in: ACM/IEEE Design Automation and Test in Europe Conf. (DATE), 2010.
[3] S. Borkar, Thousand core chips: a technology perspective, in: ACM/IEEE Design Automation Conf. (DAC), 2007.
[4] The International Technology Roadmap for Semiconductors (ITRS), 2011. <http://www.itrs.net/Links/2011ITRS/2011Chapters/2011SysDrivers.pdf>.
[5] A.Y. Yamamoto, C. Ababei, Unified system level reliability evaluation methodology for multiprocessor systems-on-chip, IEEE Int. Green Computing Conference (IGCC), 2012.
[6] Jayanth Srinivasan, Lifetime reliability aware microprocessors, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 2006.
[7] J. Shin, V. Zyuban, Z. Hu, J. Rivers, P. Bose, A framework for architecture-level lifetime reliability modeling, in: IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN), 2007.
[8] A.K. Coskun, T.S. Rosing, K. Mihic, G.D. Micheli, Y. Leblebici, Analysis and optimization of MPSoC reliability, J. Low Power Electr. 2 (1) (2006) 56–69.
[9] Z. Gu, C. Zhu, L. Shang, R.P. Dick, Application-specific MPSoC reliability optimization, IEEE Trans. Very Large Scale Integr. Syst. (TVLSI) 16 (5) (2008) 2008.
[10] Y. Xiang, T. Chantem, R. Dick, X.S. Hu, L. Shang, System level reliability modeling for MPSoCs, in: IEEE/ACM/IFIP Int. Conf. on Hardware/Software Codesign and System Synthesis (CODES + ISSS), 2010.
[11] J.P.G. Sterbenz, D. Hutchison, E. Cetinkaya, A. Jabbar, J.P. Rohrer, M. Scholler, P. Smith, Resilience and survivability in communication networks: strategies, principles, survey of disciplines, Comput. Networks (2010).
[12] A. Dalirsani, M. Hosseinabady, Z. Navabi, An analytical model for reliability evaluation of NoC architectures, in: IEEE Int. on-Line Testing Symposium (IOLTS), 2007.
[13] H. Elmiligi, A.A. Morgan, M.W. El-Kharashi, F. Gebali, A reliability-aware design methodology for networks-on-chip applications, in: IEEE Int. Conf. on Design and Technology of Integrated Systems in Nanoscale Era, 2009.
[14] C. Ababei, H. Sajjadi Kia, O.P. Yadav, J. Hu, Energy and reliability oriented mapping for regular networks-on-chip, in: ACM/IEEE Int. Symposium on Networks-on-Chip (NOCS), 2011.
[15] Failure Mechanisms and Models for Semiconductor Devices, JEDEC Publication JEP122E, 2009.
[16] T. Simunic, S.P. Boyd, P.W. Glynn, Managing power consumption in networks on chips, IEEE Trans. Very Large Scale Integr. Syst. (TVLSI) 12 (1) (2004) 96–107.
[17] Y. Wang, K. Ma, X. Wang, Temperature-constrained power control for chip multiprocessors with online model estimation, in: ACM/IEEE Int. Symp. on Computer Architecture (ISCA), 2009.
[18] Y. Wang, Q. Xie, A. Ammari, M. Pedram, Deriving a near-optimal power management policy using model-free reinforcement learning and Bayesian classification, in: ACM/IEEE Design Automation Conf. (DAC), 2011.
[19] P. Bogdan, S. Jain, R. Tornero, R. Marculescu, An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads, in: ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS), 2012.
[20] J. Donald and M. Martonosi, Techniques for multicore thermal management: classification and new exploration, in: ACM/IEEE Int. Symp. on Computer Architecture (ISCA), 2006.
[21] A.K. Coskun, T.S. Rosing, K. Gross, Utilizing predictors for efficient thermal management in multiprocessor SoCs, IEEE Trans. CAD Integr. Circuits Syst. (TCAD) 28 (10) (2009) 1503–1516.
[22] Y. Zhang and A. Srivastava, Adaptive and autonomous thermal tracking for high performance computing systems, in: ACM/IEEE Design Automation Conf. (DAC), 2010.
[23] P. Kumar, D. Atienza, Run-time adaptable on-chip thermal triggers, in: ACM/IEEE Asia and South Pacific Design Automation Conf. (ASP-DAC), 2011.
[24] R. Jayaseelan, T. Mitra, Dynamic thermal management via architectural adaptation, in: ACM/IEEE Design Automation Conf. (DAC), 2009.
[25] Y. Ge, Q. Qiu, Q. Wu, A multi-agent framework for thermal aware task migration in many-core systems, IEEE Trans. Very Large Scale Integr. Syst. (TVLSI) 20 (10) (2012) 1758–1771.
[26] T. Austin, D. Blaauw, T. Mudge, K. Flautner, Making typical silicon matter with Razor, IEEE Comput. 37 (3) (2004) 57–65.
[27] Z. Lu, J. Lach, M.R. Stan, K. Skadron, Improved thermal management with reliability banking, IEEE Micro 25 (6) (2005) 40–49.
[28] E. Karl, D. Blaauw, D. Sylvester, T. Mudge, Multi-mechanism reliability modeling and management in dynamic systems, IEEE Trans. Very Large Scale Integr. Syst. (TVLSI) 16 (4) (2008).
[29] S. Feng, S. Gupta, A. Ansari, S. Mahlke, Maestro: orchestrating lifetime reliability in chip multiprocessors, in: Int. Conf. on High-Performance Embedded Architectures and Compilers (HiPEAC), 2010.
[30] A. Tiwari, J. Torrellas, Facelift: hiding and slowing down aging in multicores, in: ACM/IEEE Int. Symp. on Microarchitecture (MICRO), 2008.
[31] C. Zhuo, D. Sylvester, D. Blaauw, Process variation and temperature-aware reliability management, in: ACM/IEEE Design Automation and Test in Europe Conf. (DATE), 2010.
[32] A.K. Coskun, R.D. Strong, D.M. Tullsen, T.S. Rosing, Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors, SIGMETRICS/Perform. (2009).
[33] J. Sun, A.K. Kodi, A. Louri, J.M. Wang, NBTI aware workload balancing in multi-core systems, in: IEEE Int. Symp. on Quality Electronic Design (ISQED), 2009.
[34] J.H. Stathis, Reliability limits for the gate insulator in CMOS technology, IBM J. Res. Develop. 46 (2002) 265–286.
[35] E. Wu, J. Sune, W. Lai, E. Nowak, J. McKenna, A. Vayshenker, D. Harmon, Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides, Solid-State Electron. 46 (11) (2002) 1787–1798.
[36] D.K. Schroder, J.A. Babcock, Negative bias temperature instability: road to cross in deep submicron silicon semiconductor manufacturing, J. Appl. Phys. 94 (1) (2003) 1–18.
[37] S. Zafar, B. Lee, J. Stathis, A. Callegar, T. Ning, A model for negative bias temperature instability (NBTI) in oxide and high k pFETs, in: Int. Symposium on VLSI Technology, 2004.
[38] S.R. Vangal et al., An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS, IEEE J. Solid-State Circuits 3 (1) (2007) 29–41.

[39] S. Bell et al., TILE64 – processor: a 64-Core SoC with mesh interconnect, in: IEEE SSCC, 2008.

[40] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-GHz mesh interconnect for a teraflops processor, IEEE Micro 27 (5) (2007) 51–61.

[41] B. Li, L.-S. Peh, P. Patra, Impact of process and temperature variations on network-on-chip design exploration, in: IEEE Int. Symposium on Networks-on-Chip (NOCS), 2008.

[42] 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, D.A. Wood, The gem5 simulator, ACM SIGARCH Comput. Architect. News Arch. (2011).

[43] N.L. Binkert, R.G. Dreslinski, L.R. Hsu, K.T. Lim, A.G. Saidi, S.K. Reinhardt, The M5 simulator: modeling networked systems, IEEE Micro 26 (4) (2006) 52–60.

[44] M.M.K. Martin, D.J. Sorin, B.M. Beckmann, M.R. Marty, M. Xu, A.R. Alameldeen, K.E. Moore, M.D. Hill, D.A. Wood, Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset, Comput. Architect. News (CAN) (2005).

[45] N. Agarwal, L.-S. Peh, N. Jha, GARNET: A Detailed Interconnection Network Model Inside a Full-system Simulation Framework, CE-P08-001, Princeton University, 2008.

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

[47] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, M.R. Stan, HotSpot: a compact thermal modeling method for CMOS VLSI systems, IEEE Trans. Very Large Scale Integr. Syst. (TVLSI) 14 (5) (2006).

[48] X. Li, J. Qin, J.B. Bernstein, Compact modeling of MOSFET wearout mechanisms for circuit-reliability simulation, IEEE Trans. Dev. Mater. Reliab. 8 (1) (2008) 98–121.

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

[50] M. Gebhart, J. Hestness, E. Fatehi, P. Gratz, S.W. Keckler, Running PARSection 2.1 on M5, Technical Report TR-09-32, The University of Texas at Austin, 2009.

[51] L. Shang, L.-S. Peh, A. Kumar, N.K. Jha, Thermal modeling, characterization and management of on-chip networks, in: Int. Symp. Microarchitecture, 2004.

[52] L. Yang, R.P. Dick, P.A. Dinda, G. Memik, X. Chen, HAPPE: human and application driven frequency scaling for processor power efficiency, IEEE Trans. Mobile Comput. (2013).

[53] C.-L. Chou, R. Marculescu, Designing heterogeneous embedded network-on-chip platforms with users in mind, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (TCAD) (2010).

Alexandre Yasuo Yamamoto received the B.S. degree in electrical engineering from UNICAMP, Brazil in 2010. He is currently a graduate student in the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND. His main research interests include lifetime reliability of chip multiprocessors and integrated circuits fabrication.

Cristinel Ababei received the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2004. He is an assistant professor in the Dept. of Electrical and Computer Engineering, Marquette University. Prior to that, from 2012 to 2013, he was an assistant professor in the Dept. of Electrical Engineering, The State University of New York at Buffalo. Between 2008 to 2012, he was an assistant professor in the Dept. of Electrical and Computer Engineering, North Dakota State University. From 2004 to 2008, he worked for Magma Design Automation, Silicon Valley. His research interests include design automation of systems-on-chip with emphasis on reliability and reconfigurable and parallel computing.