Scalable analytical model for reliability measures in aging VLSI by interacting Markovian agents

Davide Cerotti[1], Antonio Miele[2], Marco Gribaudo[2], Andrea Bobbio[1], Cristiana Bolchini[2]

[1]*DiSit - Università Piemonte Orientale, Alessandria, Italy;*
[2]*DEIB - Politecnico di Milano, Milano, Italy*

Abstract

Aging phenomena in VLSI are enhanced by the shrinkage in transistor dimension with consequent increase in operating temperature and current density, and are now recognized as a major cause in the reduction of the chip lifetime and reliability.

The present study proposes an analytical framework based on Markovian Agent Model (MAM), able to capture aging effects in VLSI systems, while considering at the same time the interactions between VLSI elements as a function of their local position on the chip and the dynamic redistribution of the workload with the progressive failure of components. The paper presents the MAM formalism and how a fairly general aging model can be built with this formalism. The flexibility and the effectiveness of the model are illustrated by computing performance-reliability related measures on two case studies with a different flavor: a Multi-Core System-on-Chip and a Solid state Drive.

Keywords: Modeling, reliability evaluation, VLSI aging, MCSoC, SSD

1. Introduction

Aging phenomena in semiconductor materials and devices have been recognized and studied since the early stages of the Integrated Circuit manufacturing, as documented, for instance, by the IEEE Annual International Reliability Physics Symposium (IRPS) [20] whose first venue was in 1962.

In the past decades, the aggressive advances in chip manufacturing process has led to shrink the dimension of transistors to tens of nanometers worsening several aging mechanisms that are now a major challenge in VLSI useful life [25, 36]. At the same time, the advances in technology scaling has enabled the integration of several processing elements within the same chip, with

Email address: [1]`[andrea.bobbio, davide.cerotti]@uniupo.it`, [2]`[cristiana.bolchini, marco.gribaudo, antonio.miele]@polimi.it` (Davide Cerotti[1], Antonio Miele[2], Marco Gribaudo[2], Andrea Bobbio[1], Cristiana Bolchini[2])

a considerable increase in the current densities, and in turn, in the operating temperatures. These effects have caused detrimental interaction effects among adjacent elements in the chip with considerable drawback in terms of devices' lifetime and reliability. As reported in the International Technology Roadmap for Semiconductors (ITRS) in 2011 [22], the temperature increase is the primary cause of an acceleration in device aging and wear-out phenomena, including electromigration (EM), time dependent dielectric breakdown (TDDB), negative bias temperature instability (NBTI) and thermal cycling (TC). Multi-Core Systems-on-Chips (MCSoCs) are heavily utilized in modern digital equipment at any level of complexity, and the analysis of MCSoCs reliability has been the object of several studies in recent years [40, 18, 43, 4]. The dramatic decrease of the lifetime of digital computing systems based on MCSoCs has pushed the focus on the performance and reliability of such devices. The studies have shown that the system-level design choices (e.g., mapping and scheduling, workload distribution, utilization levels) have a macroscopic impact on the chip lifetime higher than circuit-level reliability enhancing techniques [43, 35].

In the area of permanent computer storage devices, the nonvolatile storage technology based on NAND gates coupled with manufacturing scaling, has made it possible to increase the storage density and to reduce the storage cost per bit allowing the flash-memory Solid State Drives (SSDs) to compete with traditional magnetic HD technology. However, flash cells degrade at any Program/Erase (PE) cycle, until the cell is no longer readable with detrimental effects on the memory reliability and endurance [11, 34, 33, 38]. To mitigate the effect of cell degradation on the SSD lifetime various wear-leveling techniques have been implemented to balance the number of erasures on the memory blocks [12, 17] so extending the life span of the drive, although introducing mutual dependences among the memory blocks [42, 37].

This paper defines a fairly general analytical model for the quantitative evaluation of performance- and reliability-related measures in large VLSI systems affected by aging phenomena, and we apply it to two study cases. The proposed analytical model is based on the formalism of Markovian Agent Model (MAM) [5, 6, 2]. A MAM is a collection of Markovian Agents (MAs), where an MA is a finite-state stochastic model governed by an infinitesimal generator kernel that contains local transition rates and transition rates induced by the interaction with the other MAs. Furthermore, MAs are located in a defined geographical space, so that their local properties and the interaction induced mechanisms may depend on their relative positions.

The overall MAM model may combine MAs representing the VLSI elements (core or memory blocks) and their mutual interaction and MAs representing the aging mechanism and its effect on the VLSI elements. Each MA can capture physical characteristics of the VLSI element and how these characteristics are influenced by the interaction with the other elements and with the aging mechanisms. Further, the MAM can account for system-level design choices that have a considerable impact on the system endurance. A preliminary study of a MAM for a MCSoC has been presented in [1] showing how the reliability of such a kind of system is influenced by the variations of temperature and workload in caused by the redistribution of applications when a core failure occurs and by the interaction among the cores.

In this paper, we illustrate the capabilities of the MAM model on two case studies with two different flavors: a Multi-Core System-on-Chip and an SSD, the former being it a typical study case addressed with different strategies, the latter as a proof-of-concept for the proposed approach, being it a field where only little data is available. Section 2 introduces the formalism of MAMs and Section 3 shows how a general aging model based on MAMs can be built. Then two case studies follow, namely, the MCSoC case study in Section 4 and the SSD in Section 5. Section 7 gives an overview on related works and conclusions follow in Section 8.

2. Markovian Agent Model

A Markovian Agent Model (MAM) is an agent-based spatio-temporal analytical formalism suitable to describe large scale systems of interacting objects. Each of them is represented by a MA, a finite-state automata over which a transition kernel is defined. Agents are located in a space and both their local properties and mutual interactions may depend on their positions.

In a Markovian Agent Model model we are interested in computing the temporal evolution of the average numbers of agent in a given state for every location. Solution is obtained with a mean field approximation thanks to a set of ordinary differential equations. In this work, we extend conventional Markovian Agent Model by adding *reward states*: continuous variables that are not used to count the average number of agents in a given state, but to hold the values of a control parameter.

Conventional states evolves according to a transition kernel defined by two components: a *local transition matrix* and an *induced transition matrix*. The former describes the local behavior by an homogeneous Continuous Time Markov Chain (CTMC) whose values may depend on the position of the agent. The latter models the interactions of the MA with the other MAs. The evolution of the reward states is based on a drift function, which in turn depends on the average count of agents in a given state (either in the same or in neighbour locations). MAs may belong to different classes and agents in a specific class c are characterized by the same transition kernel; more details are described in [2]. In the extension used in this work, each agent class determines also the rewards and the corresponding drifts associated to the MAs.
From a graphical point of view, we represent conventional states with circles and local transitions with arcs (as in standard Markov Chain models). Reward states are represented with double circles, and drifts with double arrows, following a notations similar to the one used in Fluid Stochastic Petri Nets to denote continuous places and transitions [13]. The graphical notation is summarised in Figure 1, where two conventional states i and j, a reward state k, a transition with rate q_{ij} and a drift at rate λ_k are shown.

Let us call $\boldsymbol{\pi}_c(t; v)$ state vector of a single MA of class c at time t. It contains one component for each state: $\boldsymbol{\pi}_c(t; v) = [\boldsymbol{\pi}_c^D(t; v), \boldsymbol{\pi}_c^R(t; v)]$, with

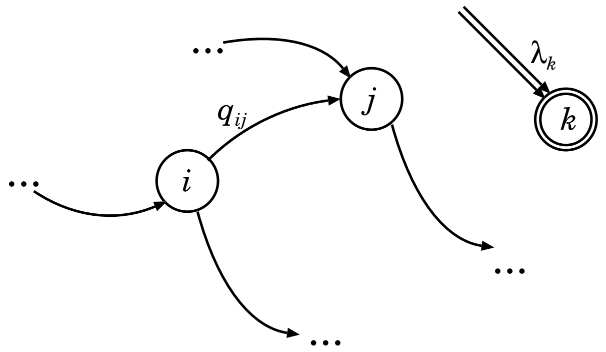


Figure 1: Graphical representation of a Markovian Agent with reward states

$\boldsymbol{\pi}_c^D(t;v)$ holding all the components corresponding to conventional states, and $\boldsymbol{\pi}_c^D(t;v)$ focusing on reward states. Elements of $\boldsymbol{\pi}_c^D(t;v)$ corresponds to the average count of agents in that state in the considered position at the given time. If we focus on a location that contains a single agent, the components of $\boldsymbol{\pi}_c^D(t;v)$ can be seen as an approximation of the state probability distribution of the agent. For reward states, the corresponding component in $\boldsymbol{\pi}_c^R(t;v)$ holds the average accumulated reward in the considered location. We also denote with $[\boldsymbol{\Pi_V}]$ the ensemble of the state vectors of all the MAs in the space at time t: in other words, a single vector that includes all the states, rewards, classes and locations of the model.

The transition kernel is a square matrix with as many rows and columns as conventional states, and it rules the stochastic behavior of an MA of class c. Its formal description is:

$$\boldsymbol{K_c}(t;v;\boldsymbol{\Pi_V}) = \boldsymbol{Q_c}(t;v) + \mathcal{I}_c(t;v;\boldsymbol{\Pi_V}) \tag{1}$$

where $\boldsymbol{Q_c}(t;v)$ is the local kernel (infinitesimal generator) that depends only on the position v of the MA, $\mathcal{I}_c(t;v;[\boldsymbol{\Pi_V}])$ is the induced kernel that depends on the position v of the MA and on the ensemble of the average state count $[\boldsymbol{\Pi_V}]$: in this way a MA can be influenced in its evolution by all the other agents (and rewards) in the entire model.
Rewards vary instead according to a drift vector $\boldsymbol{d}_c(t;v;\boldsymbol{\Pi_V})$, with one element per reward component. It can be different in any location, defined per class, and depend both on time and the ensemble of the average state count $[\boldsymbol{\Pi_V}]$.
The presence of a separated interaction component allows to avoid the construction of the whole state space and to solve the overall model by building one sub-models for each MA, and then solve them separately using the standard equation:

$$\begin{cases} \dfrac{d\,\boldsymbol{\pi}_c^D(t;v)}{d\,t} = \boldsymbol{\pi}_c^D(t;v)\,\boldsymbol{K_c}(t;v;[\boldsymbol{\Pi_V}]) \\[2ex] \dfrac{d\,\boldsymbol{\pi}_c^R(t;v)}{d\,t} = \boldsymbol{d}_c(t;v;\boldsymbol{\Pi_V}) \end{cases} \tag{2}$$

with a given initial condition $\boldsymbol{\pi}_c(0; v)$.

Thus, the solution of the overall model at any time t is obtained by solving for any MA of any class c in the space an equation of type (2). Several examples of the derivation of the structure of the kernel $\boldsymbol{K_c}(t; v; [\boldsymbol{\Pi_V}])$ without rewards are described in [6, 2].

3. Aging model in MAM

Here we will describe the general methodology used in MAM to consider the effect of aging on the stochastic behavior of the agents..

Given an agent of class c in position v we enrich the model with a reward state A that keeps track of the accumulated aging level Ψ. Such value increases at rate $\lambda(v, \boldsymbol{\Pi_V})$ that may depend on the agent location v and on the ensemble $\boldsymbol{\Pi_V}$ at time t in order to take in account the impact of other MAs to the aging of agent v. To simplify the notation, we assume that such dependency can be captured by a single parameter $\Omega = \omega(\boldsymbol{\Pi_V})$. Moreover, we assume that the agent in position v has at least a transition tr_f that depends on the aging process, typically tr_f represents a failure event. Let us call Y the random variable representing the time to failure of tr_f. Due to the dependency on the ensemble, the probability distribution of $Y(t|\Omega)$ is non-homogeneous and non-exponential.

To support such type of distribution we apply the supplementary variables approach from Cox [10]. The random variable Y is characterized by a probability density function (pdf) $f_Y(t)$ and Cumulative distribution function (Cdf) $F_Y(t)$, its hazard rate $h_Y(t)$ and accumulated hazard rate $H_Y(t)$ are defined, respectively, as:

$$h_Y(t) = \frac{f_Y(t)}{1 - F_Y(t)} \qquad H_Y(t) = \int_0^t h_Y(u)du \tag{3}$$

Assuming that the process starts at time $t = 0$, we can model its non-exponential behavior by a continuous variable Markov process, where at each point in time t, the non-exponential event modeled by Y occurs at rate[1] $\lambda_Y = h_Y(t)$. The accumulated age Ψ of the process is:

$$\Psi = H_Y(t) = \int_0^t h_Y(u)du \tag{4}$$

Since $h_Y(t)$ is a strictly positive function, $H_Y(t)$ is invertible. This allows us to express the failure rate as function of the accumulated age Ψ:

$$\lambda_Y(\Psi) = h_Y\left(H_Y^{-1}(\Psi)\right) \tag{5}$$

[1]To simplify the presentation, in this section we have deliberately not included all the dependencies in the variable: for example, we have written $\lambda_Y = h_Y(t)$ instead of $\lambda_Y(t) = h_Y(t)$

Moreover, the evolution of the accumulated age can be expressed in a differential way:

$$\frac{d\Psi}{dt} = \frac{dH_Y}{dt} = h_Y(t) = h_Y\left(H_Y^{-1}(\Psi)\right) = \lambda_Y(\Psi) \tag{6}$$

Whenever the value of Ω varies the Time To Failure (TTF) distribution of $Y(t|\Omega)$ changes, however it has been proved that the *reliability conserves* [26], that is, if at time t', the TTF distribution changes from Y to Y', we must have that:

$$R_Y(t') = R_{Y'}(t') \tag{7}$$

This is in general obtained by "time shifting" the hazard rate of the new distribution Y' to match the previously accumulated aging. However, in our model we have:

$$R_Y(t') = e^{-H_Y(t')} = e^{-\Psi} = e^{-H_{Y'}(t')} = R_{Y'}(t') \tag{8}$$

since Ψ does not depend on t or t'. In other words, the change of variable from t to Ψ that allows us to express the hazard rate as function of the aging also allows us to change directly the TTF distribution from Y to Y', conserving the reliability.

In particular for a Weibull distribution, defined by scale parameter α and shape parameter β, we have:

$$f_W(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} e^{-\left(\frac{t}{\alpha}\right)^{\beta}} \qquad F_W(t) = 1 - e^{-\left(\frac{t}{\alpha}\right)^{\beta}} \tag{9}$$

$$h_W(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \qquad H_W(t) = \left(\frac{t}{\alpha}\right)^{\beta} \tag{10}$$

$$H_W^{-1}(\Psi) = \alpha\Psi^{\frac{1}{\beta}} \qquad \lambda_W(\Psi) = \frac{\beta}{\alpha}\Psi^{\frac{\beta-1}{\beta}} \tag{11}$$

4. Case study 1: Reliability model of a Multi-Core System-on-Chip

4.1. System description

The first case study we propose in this paper shows the applicability of Markovian agents for the analysis of aging and wear-out effects in Multi-Core System-on-Chip architecture. The considered MCSoC architecture, shown in Figure 2, is composed of a set of homogeneous processing cores organized with a mesh structure. The system is used to run a multi-programmed workload, composed of various applications that are distributed on the available processing cores by the operating system according to a specific mapping policy.

Based on the workload distribution, each core may either perform some elaboration or be in an idle status, and consequently will have a specific power consumption. Moreover, based on the power spatial profile of the overall architecture, the temperature of each core is a function of the its internal power consumption and the heat conveyed by neighboring cores. Finally, the time dependent rate of wear-out of each core caused by the progressive aging is primarily related to its temperature, with an exponential form.

As a conclusion, the aging process of each unit in a MCSoC architecture is strictly related to its activity and therefore on how the workload is distributed. The complexity of this picture is even more exacerbated by considering that the failure of a core may not represent a catastrophic failure of the overall system. Indeed, the workload can be redistributed on the remaining cores thus allowing a sort of graceful degradation of the architecture. Such a strategy can be adopted till the system is able to provide the minimum required Quality-of-Service, i.e., there is a minimum number of healthy cores capable of providing the necessary computational power to complete the running workload with the required performance level.

In this scenario, the construction of the analytical model is particularly challenging because the lifetime of any single core is influenced by the operating conditions of all the other cores with a sort of "multi-step" relationship involving core utilization, power consumption, temperature and heating interference, and aging rates, as well as the possibility of the core to fail.

We here show that MAMs are suited to model the reliability of MCSoC architectures by taking into account the complex cause-effect relationship affecting core aging over the time. We will demonstrate that the MA dependence on the position in the space allows the model to investigate the effect of various workload distribution strategies on the aging of the various cores and consequently on the lifetime of the overall architecture.

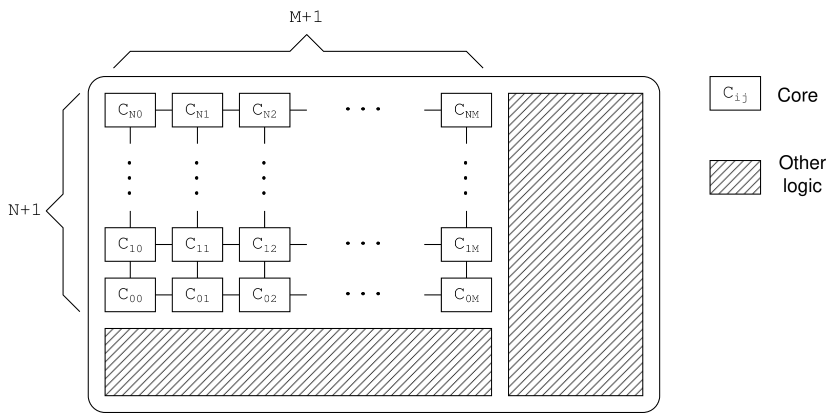


Figure 2: Simplified architecture of a many-core CPU

4.2. Thermal and aging models

The temperature model for the various cores within the complex architecture is a quite complex formulation based on differential equations involving various aspects, such as the geometrical organization of the architecture, the power characteristics of the considered technological node and the actual activity profile of each core. For this reason, we have used a set of state-of-the-art tools for characterizing a simple linear regression model capable of modeling the steady-state temperature for each core in a given activity profile of the overall architecture. The details of such characterization can be found in [1].

The obtained model takes in input $\mathcal{W}$, a binary matrix whose element $w_{i,j}$ in position (i, j) is 1 if the core in position (i, j) is in the W state, and 0 otherwise. The model returns the temperature $T_c(i, j)$ of a core in position (i, j) as:

$$\begin{aligned} T_c(i,j) &= c_0 + c_1 \cdot w_{i,j} + \\ &+ c_2 \cdot (w_{i+1,j} + w_{i-1,j} + w_{i,j+1} + w_{i,j-1}) + \\ &+ c_3 \cdot (w_{i+1,j+1} + w_{i+1,j-1} + w_{i-1,j+1} + w_{i-1,j-1}) \end{aligned} \tag{12}$$

where $c_0 \dots c_3$ are four thermal constants. In this work, we have considered $c_0 = 329$ K, $c_1 = 16$ K, $c_2 = 3$ K, and $c_3 = 1.5$ K.

As recognised by both academia and industry, aging mechanisms for integrated circuits modeled with the Weibull distribution, are validated in several sources, as documented by the JEDEC standard [23]. Discussion on academic works based on such results will be given in Section 7. In our case study, for a proof of concept, the electromigration aging mechanism has been considered and characterized as in [4]. The shape parameter $\beta = 2$, and the scale parameter $\alpha_{EM}(T)$ is computed with the Black's equation:

$$\alpha_{EM}(T) = \frac{A_{EM}(J - J_{crit})^{-n} e^{\frac{E_{a_{EM}}}{kT}}}{\Gamma\left(1 + \frac{1}{\beta}\right)} \tag{13}$$

where A_{EM} is a material-dependent constant, J and J_{crit} are the current density and its critical value activating the phenomenon, respectively, n is empirically determined constant, $E_{a_{EM}}$ is the activation energy, K is Boltzman's constant, Γ is the Gamma function, and T is the constant worst-case temperature in Kelvin degrees. In particular, as in [4], $E_{a_{EM}} = 0.48\text{eV}$, $K = 8.61673324 \cdot 10^{-5}\text{eV/K}$ and $n = 1.1$. Finally, since other parameters are usual not disclosed by the industry, they have set according the common academic practice [18, 43, 15, 4] that consists in fitting the lifetime of a single core to a given value. In our work we have considered an Mean Time To Failure (MTTF) equal to 10 years for a single core working at the constant steady-state temperature of 60°C; thus, $J = 1.5 \cdot 10^6\text{A/cm}^2$, $J_{crit} \cong 0\text{A/cm}^2$, $A_{EM} = 3 \cdot 10^5(\text{h/cm}^2)\cdot(\text{A/cm}^2)^n$.

4.3. System model

To evaluate the reliability of the overall system, we represent each core with the MA shown in Figure 3 . It has three conventional states, namely *active (W)*, *idle (I)* and *failed (F)*, used to model the activity of the core , and a reward state *Aging (A)* , used to identify the aging level of the core. The core may alternate between the W and the I modes before reaching the absorbing state F. The transition rates between W and I depend on the location of the

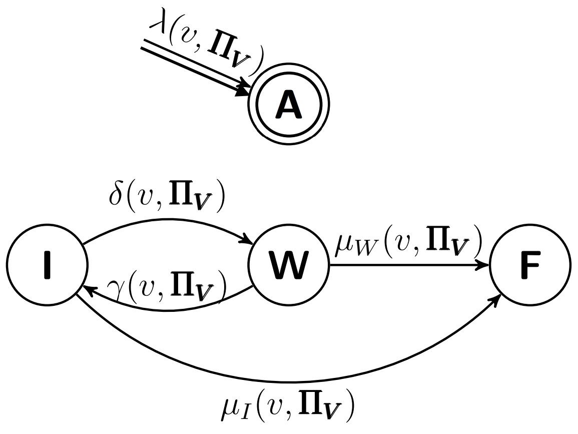

$\lambda(v, \mathbf{\Pi_V})$
A
$\delta(v, \mathbf{\Pi_V})$
I
W
$\mu_W(v, \mathbf{\Pi_V})$
F
$\gamma(v, \mathbf{\Pi_V})$
$\mu_I(v, \mathbf{\Pi_V})$

Figure 3: Markovian Agent Model of a many-core CPU core

core, on the initial workload of the core and on the policy by which the load is redistributed among the still alive cores, upon a core failure. the reason is that $\delta(v, \mathbf{\Pi_V})$ and $\gamma(v, \mathbf{\Pi_V})$ are function of the position and of the probability vectors of all the other core states (as shown in Equations from (2) to (5)). Then, the TTF either from state W or I to F, is a function of the aging level of the core that primarily depends on core temperature and utilization level.

We assume that the total workload to be processed by the system is known and constant in time. Actually, this type of workload is generally considered in embedded systems, where the same group of applications has to be repeated continuously regardless of the inputs. Moreover, such assumption might be easily removed to consider workload changes in time by adding a modulation process; we leave further studies in this direction for future work. The workload distribution policy aims to always elaborate such total workload, without exceeding a given power budget, by rotating the activation of the cores in order to stress all of them, on average, at the same level, thus with the same aging effects. We identify the average stress level of the core by means of a utilization parameter $0 \leq U \leq 1$. When a core fails, its workload is uniformly re-distributed among the remaining cores, until the cores reach an utilization $U = 1$. Subsequent core failures will cause the MCSoC to operate in a degraded condition.

The alternation between the W and I states in each core is tuned to the rotation cycle-time at which the different cores are activated in the chip. Since the transition rates $\delta(v, \mathbf{\Pi_V})$ and $\gamma(v, \mathbf{\Pi_V})$ that determine the alternation cycle between W and I are, in any case, several orders of magnitude faster than the failure rates $\mu_W(v, \mathbf{\Pi_V})$ and $\mu_I(v, \mathbf{\Pi_V})$ from W and I to F, respectively, we can assume that the states W and I reach their steady state value at the time scale of the core failure [3]. Hence:

$$U = \frac{\frac{1}{\gamma(v,\mathbf{\Pi_V})}}{\frac{1}{\delta(v,\mathbf{\Pi_V})} + \frac{1}{\gamma(v,\mathbf{\Pi_V})}} = \frac{\delta(v,\mathbf{\Pi_V})}{\delta(v,\mathbf{\Pi_V}) + \gamma(v,\mathbf{\Pi_V})} \tag{14}$$

As the time proceeds, the average number of still alive cores $n_{al}(t)$ at time t can be calculated from:

$$n_{al}(t) = n_0 - \sum_{i=1}^{n_0} \pi_F(t; v_i)$$

where n_0 is the initial number of cores in the MCSoC , v_i is the position of the core of index i and $\pi_F(t; v_i)$ is the failure probability of the core in position v_i at time t. To preserve the global workload of the chip, the utilization of the cores increases in time according to the following law:

$$U(t) = \frac{n_0}{n_{al}(t)} U_0 \tag{15}$$

where U_0 is the initial utilization of the system with n_0 cores. We assume a fixed mean active-idle cycle length L, with:

$$L = \frac{1}{\delta(v,\mathbf{\Pi_V})} + \frac{1}{\gamma(v,\mathbf{\Pi_V})} = \frac{\delta(v,\mathbf{\Pi_V}) + \gamma(v,\mathbf{\Pi_V})}{\delta(v,\mathbf{\Pi_V})\,\gamma(v,\mathbf{\Pi_V})} \tag{16}$$

Equation (16), combined with Equations (15) and (14), allows us to evaluate how the transition rates from W to I change in time:

$$\begin{cases} \gamma(v,\mathbf{\Pi_V}) & = & \dfrac{1}{L\,U(t)} \\ \delta(v,\mathbf{\Pi_V}) & = & \dfrac{1}{L\,(1-U(t))} \end{cases} \tag{17}$$

In order to characterize the aging rates in a variable working scenario, it is necessary to consider the fact that the temperature uses to change in time and, unfortunately, the basic formulation in Equation 13 receives as input a single, constant temperature value. Therefore, as show in [1], $\lambda(A)$ formula has been modeled to support such temperature change. As a first step, if we consider a single core switching between two different temperatures T_I and T_W with the same probability, asymptotically $\lambda(A)$ can be modeled as:

$$\lambda(A) = \frac{\beta}{2} A^{\frac{\beta-1}{\beta}} \cdot \left(\frac{1}{\alpha_{EM}(T_I)} + \frac{1}{\alpha_{EM}(T_W)} \right) \tag{18}$$

After that, we had also to consider the fact that, based on the thermal model, the temperature of each core depends on all its neighbors, thus sensibly increasing the complexity of Equation (18) since we have to consider the probability of each neighbor to be in each single status. Therefore, we simplified the thermal model to compute the average temperature of each core by considering the average utilization of each neighbor (which in turn asymptotically represent the probability for each core to be in a specific state). therefore, let us define sum of the utilization of the side ($U_S(i,j)$) and corner ($U_C(i,j)$) neighbors as:

$$\begin{aligned} U_S(i,j) &= \pi_W(t;v_{i+1,j}) + \pi_W(t;v_{i-1,j}) + \\ &+ \pi_W(t;v_{i,j+1}) + \pi_W(t;v_{i,j-1}) \quad (19) \\ U_C(i,j) &= \pi_W(t;v_{i+1,j+1}) + \pi_W(t;v_{i+1,j-1}) + \\ &+ \pi_W(t;v_{i-1,j+1}) + \pi_W(t;v_{i-1,j-1}) \quad (20) \end{aligned}$$

We approximate the $T_W(i,j)$ and $T_I(i,j)$ temperatures in the W and I states as:

$$\begin{aligned} T_W(i,j) &= c_0 + c_2 \cdot U_S(i,j) + c_3 \cdot U_C(i,j) \\ T_I(i,j) &= c_0 + c_1 + c_2 \cdot U_S(i,j) + c_3 \cdot U_C(i,j) \end{aligned}$$

And finally, we define $\mu_W(v_{i,j}, \mathbf{\Pi_V})$, $\mu_I(v_{i,j}, \mathbf{\Pi_V})$ and $\lambda(v_{i,j}, \mathbf{\Pi_V})$ as:

$$\begin{aligned} \mu_W(v_{i,j}, \mathbf{\Pi_V}) &= \big(\pi_A(t;v_{i,j})\big)^{\frac{\beta-1}{\beta}} \frac{\beta}{\alpha_{EM}\big(T_W(i,j)\big)} \\ \mu_I(v_{i,j}, \mathbf{\Pi_V}) &= \big(\pi_A(t;v_{i,j})\big)^{\frac{\beta-1}{\beta}} \frac{\beta}{\alpha_{EM}\big(T_I(i,j)\big)} \\ \lambda(v_{i,j}, \mathbf{\Pi_V}) &= \mu_W(v_{i,j}, \mathbf{\Pi_V})\pi_W(t;v_{i,j}) \\ &+ \mu_I(v_{i,j}, \mathbf{\Pi_V})\big(1-\pi_W(t;v_{i,j})\big) \end{aligned}$$

Basically, the age accumulates following Equation (18) where the temperature considers the effect of the local W and I states, plus the average contribution of the neighbors. However, the failure rates in the two W and I states, consider only the failure time distribution specific for the corresponding operational temperature.

4.4. Results

Let us consider a system with n_0 cores, each of them with an utilization of U_0, as introduced in Section 4.3. When a core fails, the utilization of the remaining ones is increased by the workload distribution policy to maintain a fully operational state thus preserving the total system workload of n_0U_0.

However, when all cores reach the full utilization, further failures lead the system to a degraded operating state. To analyze such behavior, let us define the workload $Wl(t)$ at time instant t as:

$$Wl(t) = \sum_v \pi_W(t;v), \quad (21)$$

where the sum of the utilisations of the cores corresponds to the current processing level of the CPU. Then, the *mean time to degradation MTTD* of the system, i.e. the first time instant at which the global workload goes below n_0U_0 is computed as:

$$MTTD_{sys} = \inf\left\{t \in [0,+\infty] : Wl(t) < n_0U_0\right\}, \quad (22)$$

Such performance index allows us to evaluate the effectiveness of the workload distribution policy in extending the fully operational phase of the MCSoC .

To evaluate the proposed MAM model, in the following paragraphs we present two experimental sessions using two different workload distribution policies. All the experiments were performed on a personal computer with an Intel Core i5-2450M CPU at 2.50GHz and 6 GB RAM with completion times of a few minutes.

Scenario 1. In the first scenario we assume a uniform distribution of the workload over a system with 16 cores and an initial per-core utilization of $U_0 = 0.95\%$, for a total workload $\Theta = 16 \cdot 0.95 \simeq 15$. In such a case due to the high utilization level of each core, we expect a short fully operational phase.

An equal amount of workload can be elaborated by systems with a larger number of cores reducing their stress level. Indeed, we can recompute the per-core utilization, required by an architecture with n cores to elaborate the workload Θ, as $U_0^n = \Theta/n$. For instance, the per-core utilization required for a system with 36 cores will be reduced to $U_0^{36} = 0.416\%$. Figure 4 shows in detail the behavior of one core of such system. After an initial short transient phase, the core reaches its fully operational condition working at the target utilization U_0^{36}. As the time goes on, the failure probability increases thus, to preserve the desired total workload, the distribution policy must increase the utilization of the cores until the value of 1 is reached. After such time, further failures lead the system to work in a degraded condition.

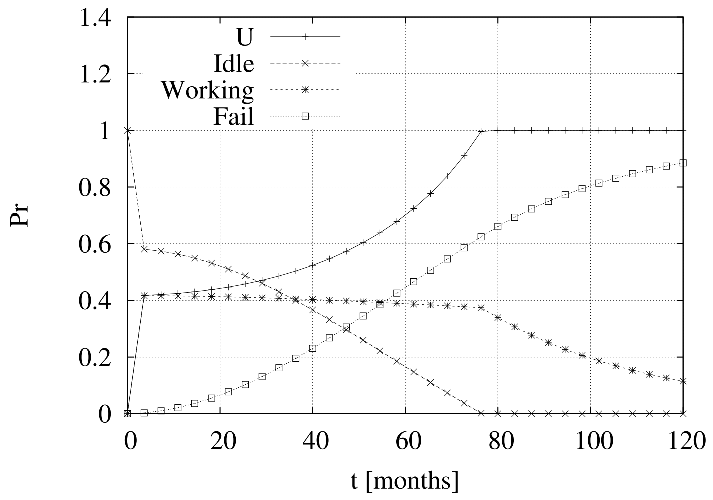


Figure 4: The utilization and state probabilities evolution in a system with 36 cores.

Figure 5 shows the behavior of the workload normalized with respect to Θ for the 16, 36 and 64 architectures. As expected, we can observe how performing the constant workload of the 16 cores chip over architectures with a greater number of cores increases the system $MTTD$. Indeed, the 16 core chip on average lasts in full operating condition for about one year, whereas the 36 CPU lasts slightly more than six years and the 64 chip more than ten years.

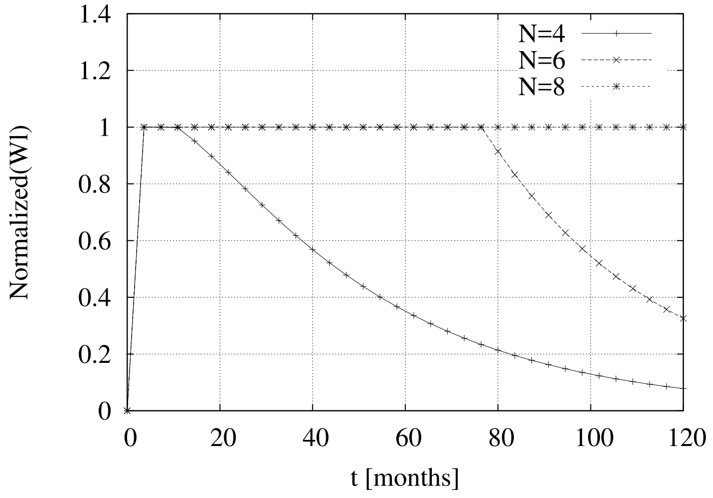


Figure 5: The global workload trend for a chip with 16, 36 and 64 cores.

Scenario 2. In such scenario the cores are partitioned into two sub-classes: initially the primary cores are active, whereas the spare cores are inactive. We consider a 12×12 CPU with 144 cores and we study two different workload redistribution policies:

- P_1: the workload lost due to the failures of the primary cores is evenly redistributed to the spare ones only, increasing their utilization level.
- P_2: the workload lost is evenly redistributed to the spare ones until the workloads of the primary and spare cores are balanced. Then, the lost workload will be redistributed evenly on all the remaining cores.

Moreover we consider different spatial distributions of the initially active/spare cores, according to the patterns shown in Figure 6.

Figure 7 shows the results for two different target utilization levels (of the initially active cores) $U_0 = 0.6$ and $U_0 = 0.9$ for the considered policies P_1 and P_2. The two policies obtain very similar results, even if P_2 performs slightly better. The pattern distribution has instead a more sensible impact: best results are obtained when the initially working cores are concentrated in a corner of the CPU. It is interesting to note that its dual - when the cores initially active are distributed over an L-shaped pattern, has the worst performances.

To better show the evolution of the system, Figure 8 presents the thermal distribution of the temperature over the chip at three different time instants for policy P_1, target $U = 0.9\%$ and initial core distribution $B3$ of Figure 6. At the beginning of the life of the system, the active cores, which are concentrated

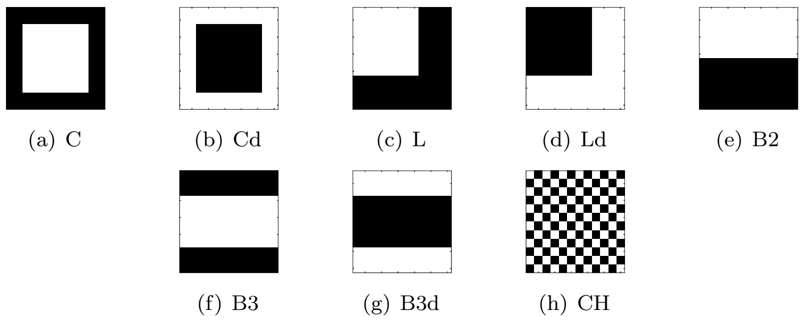


Figure 6: Primary vs spare core patterns. The primary cores are white, the spare black.

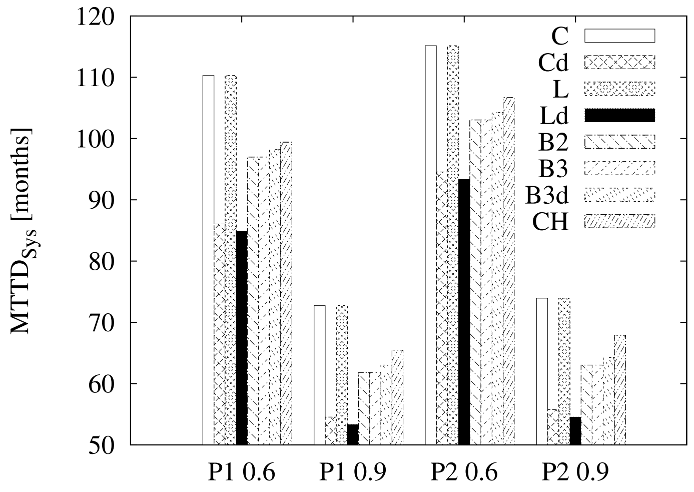


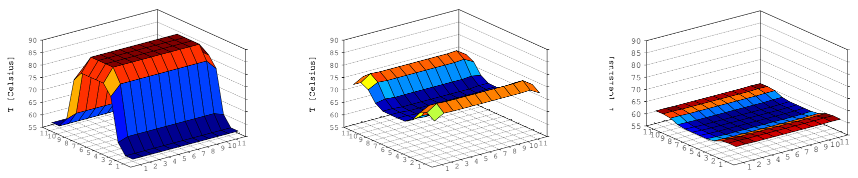
Figure 7: Comparison of the system MTTD achieved by different core patterns varying the load redistribution policy and the initial primary core utilization.

in the middle band, are working at a high temperature, while the outer bands are colder. When the MTTD is reached at around five years (Figure 8b), the CPU works at a lower temperature, and the pattern is inverted: indeed the spare cores at the beginning now have to take the place of the failed ones. After ten years, due to the distribution of the failures over the CPU, the differences between the temperature of the hot and cold area are reduced, and the surviving cores work at a much lower average temperature.

5. Case study 2: Reliability model of an SSD

5.1. System description

The second example is given to support the validity of the approach based on MAM to cope with aging phenomena in VLSI. The diffusion of SSDs is rapidly



(a) One hour. (b) Five years. (c) Ten years.

Figure 8: Average core temperature in $B3$ configuration at: a) $t = 1$ hour, b) $t = 5$ years and c) $t = 10$ years.

increasing, even if the major critical point remains the endurance and reliability due to wear-out. Although the model in not formally validated, and results are not compared with measurement, all the parameters and effects have been defined to match known behaviour and physical characteristics described in the available literature. SSDs are the current alternative to traditional hard disk drives (HDD) where rotating platters are replaced by NAND based memories. They are non-volatile storage devices, generally composed by a controller and one or more solid state memories [33]. SSDs use the same interfaces and form factors of traditional HDD, so that they can easily fit in commodity infrastructures. Data is stored into array of NAND cells that are organised into *pages* and *blocks*. Pages are the smallest unit that can be read or written in a single command. They are usually from 4KB to 32KB of size, and are grouped into blocks, generally consisting of $N_{P \times B} = 64$ to $N_{P \times B} = 512$ pages each, for a total of 256KB to 16MB. For technical reasons, only empty pages can be written, which must be erased before being updated. However, erasing can only occur at the block level (i.e. group of $64 \sim 512$ pages), destroying all of its content. A consequence of the particular read/write/erase structure of SSD is that pages can be in three states: *Empty*, they do not contain data; *Used*, they store useful data; *Dirty*, they contain invalid data.

As introduced, only empty pages can be written, and when the OS deletes a file, its block is marked as dirty with a special command called TRIM. A page cannot be overwritten without erasing the entire block first. Only dirty (and empty) pages can be erased to free space, but this must be done at the block level: as a consequence, all the pages in the block must be either dirty or empty. If this is not the case, the controller must either move, or read into memory and then write again the pages in use when erasing a block. The writes to move valid data lead to a problem called *Write Amplification Factor (WAF)*: when no more empty pages are available, the controller must read and write a larger number of pages to perform a single write operation. This effect has serious impact on the SSD performance and endurance. In order to limit the WAF, SSD controllers usually performs some sort of Garbage Collection (GC) [7, 17, 32]: they harvest for dirty pages, consolidate the ones in use, and erase blocks to create new empty pages.

When focusing about reliability [33], SSDs have a smaller error rate compared to traditional HDD, since they do not involve mechanical parts. However they suffer from severe aging problems. In particular, each block can support a limited number of program/erase (PE) cycles, after which it becomes unusable. For these reasons, SSD are characterised by a maximum amount of data that can be written on a disk before all its pages reaches the maximum number of PE cycles. Not all pages are erased and rewritten with the same rate, so it is quite common that some pages wear out faster than other ones. To mitigate this effect, SSDs usually apply a *wear levelling* algorithm to balance the utilisation of all the pages.

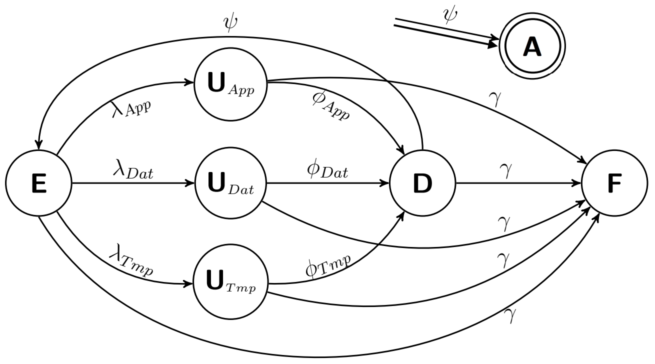


Figure 9: A Markovian Agent modeling the pages in a block of a SSD.

5.2. System model

The SSD has N_B blocks of N_P pages each. We model the time behaviour of a single block with a MA composed by 6 conventional states that model the the evolution of pages inside a single block. A reward state models the aging of the pages in the block. The complete MA is reported in Figure 9. In this case, to simplify the notation, dependencies of the rates have not been shown in the picture.

The conventional states of the MA have the following meaning. State `E` denotes empty pages, state $\mathtt{U}_x$ is split in 3 states ($\mathtt{U}_{App}$, $\mathtt{U}_{Dat}$ and $\mathtt{U}_{Tmp}$) that denote pages in use, state `D` denotes dirty pages and state `F` failed pages.

The number of pages in block v ($v = 1, \ldots, N_B$), in state `X` (`X`= `E`,$\mathtt{U}_{App}$, $\mathtt{U}_{Dat}$, $\mathtt{U}_{Tmp}$,`D`,`F`) at time t and the total number of pages in state X at time t in the SSD are denoted by, respectively $\pi_X(t, v)$, with $\sum_{X \in \Omega} \pi_X(t, v) = N_P$, being Ω the state space of the conventional components of the MA, and $\boldsymbol{\pi}_X(t) = \sum_{v=1,N_B} \pi_X(t, v)$.

Initially, each block starts with all the pages in the empty state ($\pi_E(t, v) = N_P$). A writing request moves a page from the empty state `E` to one of the in-use states $\mathtt{U}_x$. Since writing requests may have very different characteristics, we model a workload composed by three components to make the model flexible enough and adaptable to different realistic situations. The first component models the access to pages containing the operating system code and the main applications that are seldom written and seldom modified, and are represented by state $\mathtt{U}_{App}$; the second component models pages that are written and deleted at almost constant rate (i.e., temporary files) during the lifetime of the SSD and are represented by state $\mathtt{U}_{Tmp}$; the third component models pages that are basically written only after the initial setup has been completed, and represents the main data a user saves on her disk. Such data is deleted only when available disk space starts becoming an issue and is represented by state $\mathtt{U}_{Dat}$. Pages of type x (with $x \in \{App, Dat, Tmp\}$) arrive at the SSD at a rate $\lambda_x(t)$ and are assigned to a block v according to an assignment function $l_x(v; [\Pi_V])$, so that : $\lambda_x(t; v; [\Pi_V]) = \lambda_x(t) \cdot l_x(v; [\Pi_V])$.

Writing rates $\lambda_x(t; v; [\Pi_V])$ depend on the type of page, on the elapsing of time (to reflect that the writing rate may change during the life of the memory), and on the entire state of the MAM (to support aging and wear-leveling algorithms). Pages are deleted at rates $\phi_x(t)$. Deletion simply marks pages as dirty and is represented in the BMA by the transitions from $\mathtt{U}_x$ to D. A deleted page becomes invalid and cannot be overwritten. Mixing the relative consistency of the $\mathtt{U}_x$ states with the write and delete rates, we can model a very wide variety of workload conditions.

Dirty pages cannot be overwritten, but can be erased since contain invalid data. When a block is full just of empty or dirty pages, it can be erased. Erasing can also occur when there are no longer empty pages on the disk but still dirty pages. In this case, the SSD dumps the used pages from the block into the main memory, erases the entire block, and finally rewrites the old and the new data. Erasing events are modeled by the transition that connects states D to state E, occurring at rate $\psi(t; v; [\Pi_V])$. Garbage collection erases dirty pages in a block to increase available free space [7].

Blocks age at each PE cycle, and can be erased only for a limited number of times: reward state A counts the number of erasures in a block. State A is characterized by a drift $\psi(t; v; [\Pi_V])$ that matches the transition from state D to state E. In this way, $\pi_A(t; v)$ measures the cumulative number of erased pages in block v up to time t. A page may fail from any state X ($\mathtt{X} \notin \mathtt{F}$) to F at a rate $\gamma(t; v; [\Pi_V])$ that depends on the accumulated age of block v.

In particular, in this scenario $\gamma(t; v; [\Pi_V])$ has been chosen to match the hazard rate of a Weibull distribution according to Equation (10), where the time variable has been replaced by the value of reward state A, $\pi_A(t; v)$:

$$\gamma(t; v; [\Pi_V]) = \frac{\beta}{\alpha} \left(\frac{\pi_A(t; v)}{\alpha} \right)^{\beta - 1} \tag{23}$$

In Equation (23), β and α are the shape and scale parameters, respectively.

5.3. Results

We derive a set of numerical experiments on an illustrative example of an SSD composed by $N_B = 48$ blocks, each one containing $N_P = 64$ pages. The aim of the numerical example on a small-scale case is to show the capabilities and potentialities of the model in a wide range of application cases; scalability of the model will be discussed in Section 6.

Page occupation is determined by the arrival rates $\lambda_x(t)$ and by the assignment function $l_x(v;[\Pi_V])$ that assigns incoming requests to blocks and pages. To gain flexibility in the input conditions and make them time dependent, the rates $\lambda_{App}(t)$, $\lambda_{Dat}(t)$ and $\phi_{Dat}(t)$ are determined by the following function $\rho(t, y_0, y_1, t_0, \sigma)$:

$$\rho(t, y_0, y_1, t_0, \sigma) = y_0 + (y_1 - y_0)\frac{\tan^{-1}\left(\frac{t-t_0}{\sigma}\right) - \tan^{-1}\left(-\frac{t_0}{\sigma}\right)}{\frac{\pi}{2} - \tan^{-1}\left(-\frac{t_0}{\sigma}\right)} \tag{24}$$

where $y_0 = \lim_{t\to-\infty} \rho(t, y_0, y_1, t_0, \sigma)$, $y_1 = \lim_{t\to+\infty} \rho(t, y_0, y_1, t_0, \sigma)$, t_0 is the time at which the function reaches the mean value $\rho(t, y_0, y_1, t_0, \sigma) = (y_0+y_1)/2$ and σ is the slope. In this way, we can modulate the time interval in which the rates are active, their absolute value and the variation of the absolute value in time.

In particular, we assign the following values to the parameters in Equation (24). $\lambda_{App}(t) = \rho(t, 40, 0.001, 10, 2)$, $\lambda_{Dat}(t) = \rho(t, 1, 25, 15, 5)$ and $\phi_{Dat} = \rho(t, 0.0156, 0.1875, 400, 10)$ (see Figure 10). The other arrival and deletion rates are considered to be constant and set to: $\lambda_{Tmp} = 8$ req./day, $\phi_{Tmp} = 1/8$ req./day and $\phi_{App} = 1.56\dot{1}0^{-5}$ req./day. With a shape parameter $\beta = 40$ and scale parameter $\alpha = 800$ in the Weibull distribution of Equation (23), the failure rate becomes an increasing function of the age.

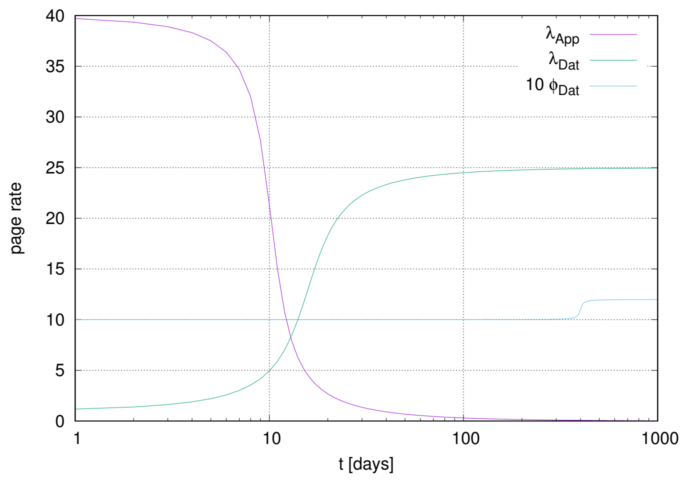


Figure 10: Arrival and deletion rates as function of time.

Blocks are assigned in order of their increasing address with incoming requests occupying the first available block, that is:

$$l_x(v;[\Pi_V]) = \begin{cases} 1 & \text{if } \pi_E(t;v') = 0 \ \forall\ v' < v \land (\pi_E(t;v) > 0) \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

To simplify the solution of the model, we suppose that garbage collection is triggered every Δ_{GC}, and operates when the number of dirty pages in a block exceeds a threshold χ. All blocks with $\pi_D(t;v) > \chi$ are erased generating valid space but, at the same time, increasing the age of the block. The GC algorithm is formulated in Equation (27).

$$\begin{aligned}
&\text{if } \pi_D(t;v) > \chi \\
&\quad \text{if } t = n \cdot \Delta_{GC},\, n \in \mathbb{N}: \\
&\qquad \pi_E(t^+;v) = \pi_E(t^-;v) + \pi_D(t^-;v) \\
&\qquad \pi_D(t^+;v) = 0 \\
&\qquad \pi_A(t^+;v) = \pi_A(t^-;v) + \pi_D(t^-;v)
\end{aligned} \tag{26}$$

The procedure defined in Equation (26) assumes that the speed at which pages are erased is performed in negligible time with respect to the time of the other actions included in the model, and mathematically corresponds to setting function ψ to a Dirac's delta whenever the conditions are verified and zero otherwise. The numerical solution procedure implements the proposed technique by solving the differential equation (Equation (2)) for time steps of size Δ_{GC}, and then applying Equation (26) to define the initial condition of the next time step. The threshold used in the evaluation is $\chi = 8$ pages, and the time interval is $\Delta_{GC} = 20$ days.

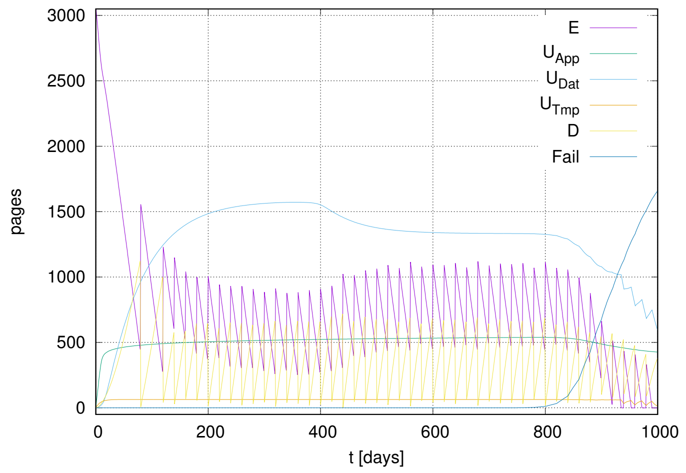


Figure 11: Temporal evolution of the total number of pages in each state.

Figure 11 shows the evolution of the total number of pages in each state all over the disk (i.e. $\boldsymbol{\pi}_X = \sum_v \pi_X(t^-;v)$ with $X \in \{E, D, F, U_{App}, U_{Dat}, U_{Tmp}\}$). Initially, all pages are empty and then the three considered types of block start filling the disk. The first time condition $\pi_D(t;v) > \chi$ of Equation (27) is satisfied occurs at $t = 80$ days: the selected time interval Δ_{GC} makes the depicted line having a saw-tooth shape in the empty and dirty pages. The number of failed pages remains negligible until around $t = 800$ days, when the wear-out effect becomes dominant, and pages rapidly start failing, making the disk not usable.

The distribution of the number of pages per block in the different states with respect to time is shown in Figure 12. Several insights can be highlighted. Figures 12a) and 12b) show the number of pages per block in states `E` and `D`, respectively. The considered assignment policy leaves some of blocks almost unused (i.e., those with index $v > 40$), and exploits them only when the disk is close to is failure (Figure 12a). Application pages occupy the initial blocks of the SSD (Figure 12c) (i.e., with index $v < 10$), while the other types of pages occupy the remaining usable space of the disk in an almost uniform way (Figure 12d-e). Failures then occur prevalently in the area where user data and temporary files are stored (Figure 12f).

The system aging is shown in Figure 13 by reporting for each block the cumulative number erased pages in state `A`, $\pi_A(t; v)$. As expected, blocks with a low number of PE cycles like those that are almost not used ($v > 40$) or those that contain mostly application data ($v < 10$) have a low level of wearing. Data blocks or temporary blocks present an almost linear aging with time. The increasing failure rate of the considered Weibull distribution makes them fail when they approach the considered scale parameter α.

5.4. Wear leveling

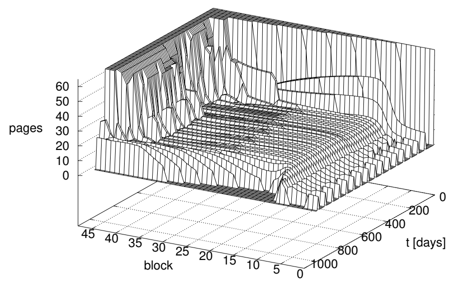
Wear leveling algorithms transfer pages from one block to another to obtain a more uniform distribution of the age among blocks, thus increasing its endurance.

In this example, we introduce a simple wear leveling procedure inspired by the *age convergence* mechanism proposed in [30]. Every Δ_{WL}, the system swaps the K_{WL} most aged blocks with the K_{WL} least aged ones. In more details, the wear leveling procedure searches the subset $V_M \in V$ of K_{WL} blocks for which the age $\pi_A(t; v)$ is maximum, and the subset $V_m \in V$ of K_{WL} blocks for which the age $\pi_A(t; v')$ is minimum, then swaps any block $v \in V_M$ with a block $v' \in V_m$. Formally, the algorithms can be described as follows:

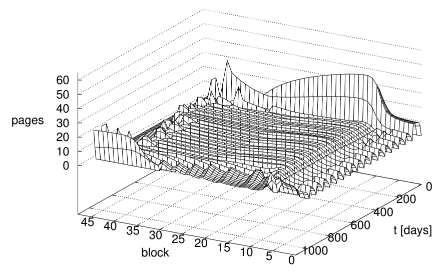
$$\begin{aligned}
&\text{if } t = n \cdot \Delta_{WL}, n \in \mathbb{N} :\\
&\quad V_M = \{K_{WL} \text{ blocks } v, \text{ with maximum } \pi_A(t^-; v)\}\\
&\quad V_m = \{K_{WL} \text{ blocks } v', \text{ with minimum } \pi_E(t^-; v')\}\\
&\text{for any state } X = E, U_{App}, U_{Dat}, U_{Tmp}, D \text{ and } v \in V_M \text{ wand } v' \in V_m\\
&\quad \pi_X(t^+; v) = \pi_X(t^-; v'), \text{ with } v \in V_M \text{ and } v' \in V_m \qquad (27)\\
&\quad \pi_X(t^+; v') = \pi_X(t^-; v), \text{ with } v \in V_M \text{ and } v' \in V_m
\end{aligned}$$

This procedure assumes that block swapping takes a negligible amount of time with respect to the other time scales of the model. As a further simplifying assumption, we consider that wear leveling has not a relevant impact on the aging of the blocks being swapped.

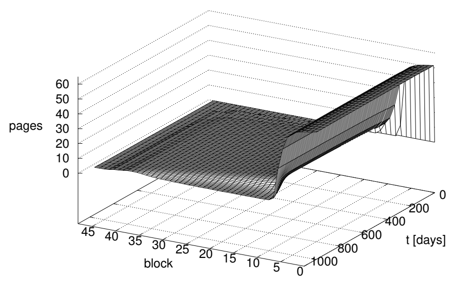
Figure 14 shows the evolution of the total number of pages in a given state when wear leveling is applied. As it can be seen, the model behaves exactly as



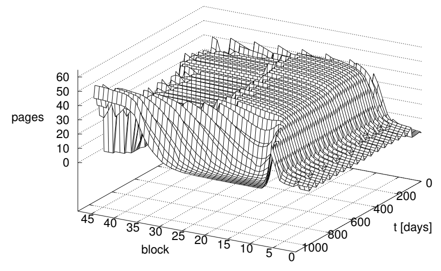
a) Empty



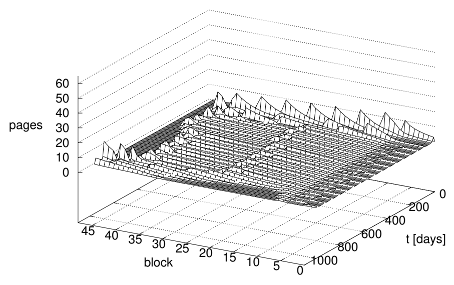
b) Dirty



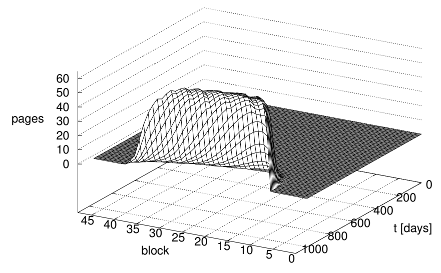
c) Used by apps



d) Used by data



e) Used by temp



f) Failed

Figure 12: Temporal evolution of the states of the pages.

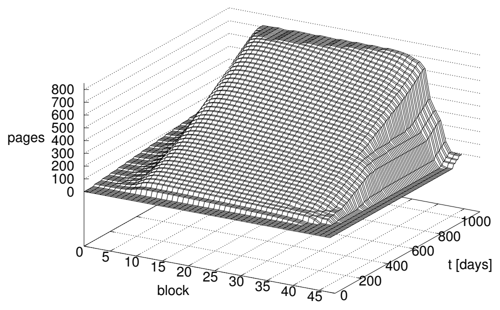


Figure 13: Temporal evolution of the age of the pages.

the one shown in Figure 11, yet the disk starts deteriorating at $t = 1000$ days, showing a 25% improvement in the expected lifetime of the SSD.

Figure 15 shows the distribution of aging among the blocks. Wear leveling has spread aging more uniformly over the disk leading to a better utilization of the blocks. In this case, when failures occur, they are evenly distributed over the disk, as shown in Figure 16.

6. Complexity of the algorithms

Multi-Core System-on-Chip. The state space of the case study in Sec. 4 has the following size. Each of the $n_0 = 144$ cores is represented by the MA in Figure 3 composed by a total of $N_S = 4$ states. The state space of the whole system is obtained by the Cartesian product as:

$$N_T = {N_S}^{n_0} = 4^{144} \approx 5.0 \cdot 10^{86}$$

which is a number that is hardly explored even by simulation. Our analytical technique is feasible because we solve separately 144 equations of the form (2), thus reducing an exponential complexity to a linear one. However, to achieve such result we need to compute for each MA in position v the induced kernel $\mathcal{I}(t; v; \mathbf{\Pi_V})$ that accounts for the influence of other agents on the behavior of the MA in position v. When all MAs influence each others according to their current states, the computation of the induced kernel requires $\mathcal{O}({n_0}^2 {N_S}^2)$ operations. Such complexity is greatly reduced when the interactions of each MA are limited to a restricted region around its position and to specific states of the agents. In the simplified thermal model of the MCSoC example, the temperatures of the cores in the W and I states are influenced only by the utilizations of the side and corner neighbors, as computed in Eqs. (19) and (20). In this way the induced kernel computation complexity is $\mathcal{O}(n_0 n_I)$, where $n_I = 8$ is the maximum number of neighbors of an MA. Equation (2) is solved by using standard numerical techniques over a dicretized time interval $0 \ldots T_h$. Such

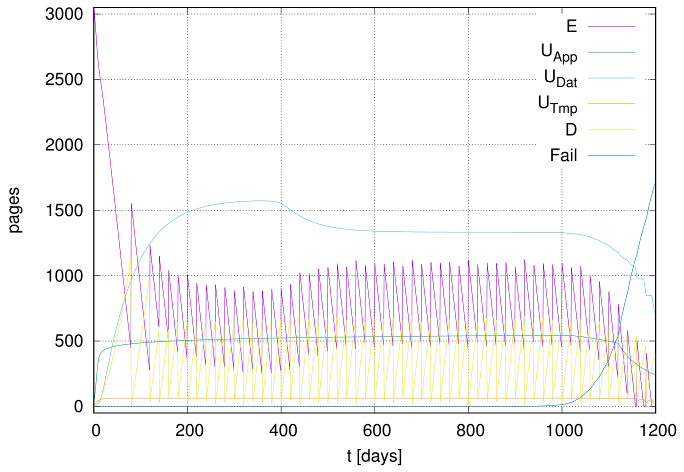


Figure 14: Temporal evolution of the total number of pages in each state with wear leveling algorithm.

interval is discretized in $k_0 = \lceil T_h/\Delta t \rceil$ time points, where Δt is the size of the integration step. The complexity of the algorithm is thus $\mathcal{O}(k_0(n_0 N_I + n_0 N_S))$ because for each time point and each MA is required to compute the influence of its N_I neighbors and to solve an equation of form (2) with a state vector $\boldsymbol{\pi}(t; v)$ of size N_S. We can observe that limiting and confining the interactions between MAs is the principal way to reduce the complexity of the proposed technique.

Solid State Drive. In the example of Sec. 5, the $0 \dots T_h$ time horizon of the analysis is uniformly discretized with steps of size Δ_{GC} yielding $k_1 = \lceil T_h/\Delta_{GC} \rceil$ time points. For each point and for each block v in isolation: i) the evolution of the block during an interval of size Δ_{GC} is computed, then ii) if the resulting number of dirty pages in the block is greater than the threshold χ, the pages are updated according to Equation (26). Phase i) requires the computation of equation (2) with a state probability vector $\boldsymbol{\pi}_c(t; v)$ of size $N_S = 7$. The equation is solved over a time interval of size Δ_{GC} discretized in $k_2 = \lceil \Delta_{GC}/\Delta t \rceil$ points. The complexity of phase i) for a single block is $\mathcal{O}(k_2 N_S)$, the complexity of phase ii) is $\mathcal{O}(1)$ for a total of $\mathcal{O}(N_B N_S k_1 k_2)$=$\mathcal{O}(N_B N_S T_h/\Delta t)$ operations for the analysis of the whole system. In presence of the wear leveling algorithm $k_3 = \lceil T_h/\Delta_{WL} \rceil$ block swapping operations are further required, which consist in finding the K_{WL} youngest and the K_{WL} oldest blocks and exchanging their pages as in Equation (27). Searching such blocks can be computed using an heap data structure with $\mathcal{O}(K_{WL} log N_B)$ operations, whereas the page exchanges take $\mathcal{O}(N_S)$, thus a total of $\mathcal{O}(k_3(K_{WL} log N_B + N_S))$ operations are additionally required. Even with the addition of the wear leveling operations, the resulting complexity is linear with respect to the number of blocks and the number of states, moreover increasing the number of pages per block has no impact on the complexity with a clear benefit on the scalability of the approach.

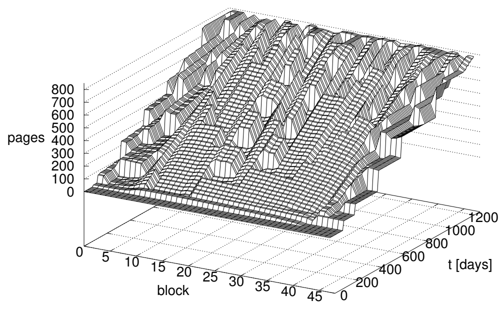


Figure 15: Temporal evolution of the age of the pages with wear levelling algorithm.

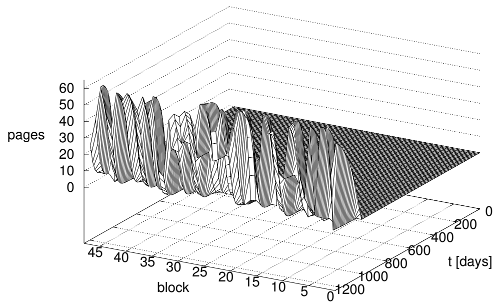


Figure 16: Temporal evolution of the failure for the pages with wear leveling algorithm.

However, care is needed during the solution of the differential equations due to the arising of *stiff* $\boldsymbol{K_c}(t; v; [\boldsymbol{\Pi_V}])$ matrices. In such a case, with basic resolution techniques a very small Δt is required to keep the stability of the analysis, thus largely increasing the solution time. To overcome this problem, advanced multi-step and adaptive-stepsize techniques, such as Runge-Kutta methods, are required to achieve a satisfying trade-off between accuracy and efficiency of the solution.

7. Related works

Aging phenomena in integrated circuits have been widely investigated in the literature; we will here discuss system level models and approaches for MCSoCs . The industrial standard procedure for the estimation of the lifetime of a class of devices has been proposed in [23] and is based on post-production accelerated aging experiments. Such models generally consider the chip as a single unit, thus considered damaged after the first failure, and assume a fixed worst-case operating temperature. These two aspects represent a considerable limitation in the early estimation of the lifetime of a modern computing system. Our proposal allows considering the temporal evolution of the system, providing a more detailed analysis than the worst case and models subsequent core failures. The first dynamic reliability management strategy acting at system-level has been proposed in [39]. The work combines the aging models defined in [23] with the sum-of-failure-rates (SOFR) approach to take into account several aging mechanisms. However, it considers a single-core device, and, adopts an exponential failure distribution, that is not very realistic. Various subsequent works [9, 24] suffer from the same limitations. Our work instead aims at considering a large number of cores to tackle possible future scenario where the parallelism in a many-core system will grow dramatically.

Weibull or lognormal distributions have been later considered in [18, 43] to track aging history. For computability reasons, the numerical solution of such model is simplified: a subset of representative workload traces is extracted for a reduced period of time with a fine granularity; from them, the average failure rate [18, 43] is extrapolated to be considered during the MTTF computation. Here we presented a novel way to include aging, factorized when possible to avoid the explosion of the computational complexity of the model, but preserved separately for the single cores.

When considering the occurrence of multiple subsequent core failures, MCSoC are actually K-out-of-N:F systems. Such model is based on multiple integrals [28], whose dimension is given by the number of failures the system can tolerate. In practice, its analytic solution is unfeasible. Thus, many approaches [18, 24] consider the entire system not to survive beyond the first failure, while some other one [19, 43, 15, 4] adopt Monte Carlo simulation or the multi-armed bandits approach [29] to quickly estimate the multiple integrals required to study the time to failure in an analytic way. Such technique will become computationally expensive when considering a temporal redistribution of the workload to maintain a target service level: the MAM approach instead provides reasonably accurate solutions in reduced computation times.

A major concern in the widespread diffusion of NAND flash memory technology as permanent storage devices is about performance, reliability, PE endurance and data retention time [14]. Various design techniques have been investigated to achieve an acceptable bit error rate to enhance reliability and endurance and to tradeoff between reliability and performance. Error correcting codes, are systematically used at the controller level, to achieve an acceptable raw-bit error rate. Publications proposing ECC based solutions for flash memories are in [8, 44]. Page is the smallest storage unit and write/read operations are performed at page level. However, erase operations are performed at block level. GC is a process required to create erased blocks [41] at the cost of incurring in a *write amplification.* A GC algorithm which reduces the WAF by randomly selecting a number of blocks and acting on the block with the fewest number of valid pages, is described in [16, 17]. Results from analytical models are restricted to uniform random writes.

RAID architectures are essential for SSD [21, 27] to increase the reliability and endurance of these storage devices. However, traditional RAID approach may have a negative effect and specific RAID techniques must be studied in conjunction with garbage collection [31, 32].

Analytical and simulation results are obtained from synthetic workloads, but now consistent field studies from large samples of devices in real operating conditions are available as documented in [38].

8. Conclusions

In this work, we have presented a MAM for studying the reliability of a MCSoC CPU. With the proposed technique, we are able to study CPUs composed by a large number of cores, and consider complex non-homogeneous non-exponential time to failure distributions with respect to their expected lifetime because of aging phenomena. Future work will focus on different application scenarios, also trying to remove the simplifying assumption used both in the thermal and in the aging models.

Acknowledgments

This work is original and has been partially supported by EUBra-BIGSEA (690116), a Research and Innovation Action (RIA) funded by the European Commission under the Cooperation Programme, Horizon 2020 and the Ministrio de Cincia, Tecnologia e Inovao (MCTI), RNP/Brazil (grant GA-0000000650/04) and has a financial support of the Università del Piemonte Orientale.

References

[1] A. Bobbio, C. Bolchini, D. Cerotti, M. Gribaudo, and A. Miele. Scalable analytical model of the reliability of multi-core systems-on-chip by interacting markovian agents. In EAI Int. Conf. on Performance Evaluation Methodologies and Tools (VALUETOOLS), pages 1–9, 2017.

[2] A. Bobbio, D. Cerotti, M. Gribaudo, M. Iacono, and D. Manini. Markovian agent models: A dynamic population of interdependent markovian agents. In E. K. Al-Begain and A. B. (Eds.), editors, Seminal Contrib. to Modelling and Simulation, pages 185–203. Springer, 2016.

[3] A. Bobbio and K. Trivedi. An aggregation technique for the transient analysis of stiff Markov chains. IEEE Trans. on Computers, C-35:803–814, 1986.

[4] C. Bolchini, M. Carminati, M. Gribaudo, and A. Miele. A lightweight and open-source framework for the lifetime estimation of multicore systems. In Proc. Int. Conf. Computer Design, pages 166–172, 2014.

[5] D. Bruneo, M. Scarpa, A. Bobbio, D. Cerotti, and M. Gribaudo. Markovian agent modeling swarm intelligence algorithms in wireless sensor networks. Performance Evaluation, 69:135–149, 2012.

[6] D. Bruneo, M. Scarpa, A. Bobbio, D. Cerotti, and M. Gribaudo. An intelligent swarm of markovian agents. In J. Kacprzyk and W. Pedrycz, editors, Springer Handbook of Comp. Intelligence, pages 1345–1359. Springer Berlin Heidelberg, 2015.

[7] L.-P. Chang, T.-W. Kuo, and S.-W. Lo. Real-time garbage collection for flash-memory storage systems of real-time embedded systems. ACM Trans. Embed. Comput. Syst., 3(4):837–863, Nov. 2004.

[8] H. Choi, W. Liu, and W. Sung. Vlsi implementation of bch error correction for multilevel cell nand flash memory. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 18(5):843–847, May 2010.

[9] A. Coskun, T. Simunic, K. Mihic, G. D. Micheli, and Y. Leblebici. Analysis and optimization of MPSoC reliability. J. Low Power Electr., pages 56–69, 2006.

[10] D. R. Cox. The analysis of non-markovian stochastic processes by the inclusion of supplementary variables. Math. Proc. of the Cambridge Philosophical Society, 51(3):433–441, 1955.

[11] Y. Deng, L. Lu, Q. Zou, S. Huang, and J. Zhou. Modeling the aging process of flash storage by leveraging semantic i/o. Future Generation Computer Systems, 32:338 – 344, 2014. Special Section: The Management of Cloud Systems, Special Section: Cyber-Physical Society and Special Section: Special Issue on Exploiting Semantic Technologies with Particularization on Linked Data over Grid and Cloud Architectures.

[12] P. Desnoyers. Analytic modeling of ssd write performance. In Proceedings of the 5th Annual International Systems and Storage Conference, SYSTOR '12, pages 12:1–12:10, New York, NY, USA, 2012. ACM.

[13] M. Gribaudo, M. Sereno, A. Horvth, and A. Bobbio. Fluid stochastic petri nets augmented with flush-out arcs: Modelling and analysis. Discrete Event Dynamic Systems: Theory and Applications, 11(1-2):97–117, 2001. cited By 42.

[14] L. M. Grupp, J. D. Davis, and S. Swanson. The bleak future of nand flash memory. In 10th USENIX Conference on File and Storage Technologies (FAST 12), page 2, 2012.

[15] A. Hartman and D. Thomas. Lifetime improvement through runtime wear-based task mapping. In Int. Conf. Hardware/software codesign and system synthesis, pages 13–22, 2012.

[16] B. V. Houdt. Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data. Performance Evaluation, 70(10):692 – 703, 2013. Proceedings of IFIP Performance 2013 Conference.

[17] B. V. Houdt. On the power of asymmetry and memory in flash-based ssd garbage collection. Performance Evaluation, 97:1 – 15, 2016. Performance Evaluation Methodologies and Tools: Selected Papers from VALUETOOLS 2013.

[18] L. Huang and Q. Xu. AgeSim: A simulation framework for evaluating the lifetime reliability of processor-based SoCs. In Conf. Design Autom. & Test in Europe, pages 51–56, 2010.

[19] L. Huang and Q. Xu. Lifetime reliability for load-sharing redundant systems with arbitrary failure distributions. Trans. Reliability, 59(2):319–330, 2010.

[20] IEEE Reliability Society. Annual International Reliability Physics Symposium. http://www.irps.org/.

[21] S. Im and D. Shin. Flash-aware raid techniques for dependable and high-performance flash memory ssd. IEEE Transactions on Computers, 60(1):80–92, Jan 2011.

[22] ITRS. Int. Tech. Roadmap for Semiconductors. http://www.itrs2.net/, 2011.

[23] JEDEC Solid State Tech. Ass. Failure mechanisms and models for semiconductor devices. JEDEC Publ. JEP122G, 2010.

[24] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge. Multi-Mechanism Reliability Modeling and Management in Dynamic Systems. Trans. on VLSI Systems, 16(4):476–487, 2008.

[25] J. Keane and C. H. Kim. Transistor aging. IEEE Spectrum, Apr 2011.

[26] D. Kececioglu. Reliab. Eng. Handbook (Vol. 1). Prentice-Hall, Upper Saddle River, NJ, USA, 1991.

[27] J. Kim, E. Lee, J. Choi, D. Lee, and S. H. Noh. Chip-level raid with flexible stripe size and parity placement for enhanced ssd reliability. IEEE Transactions on Computers, 65(4):1116–1130, April 2016.

[28] H. Liu. Reliability of a load-sharing k-out-of-n:G system: non-iid components with arbitrary distributions. Trans. Reliability, 47(3):279–284, 1998.

[29] C. Ma, A. Mahajan, and B. H. Meyer. Multi-armed bandits for efficient lifetime estimation in MPSoC design. In Proc. of Design, Automation Test in Europe Conf., pages 1540–1545, 2017.

[30] A. A. McEwan, , and I. F. Mrl. Age distribution convergence mechanisms for flash based file systems. Journal of Computers, 7(4):988–997, 2012.

[31] A. A. McEwan and M. Z. Komsul. Reliability and performance enhancements for ssd raid. Microprocessors and Microsystems, 52:461 – 469, 2017.

[32] A. A. McEwan and M. Z. Komsul. Age aware pre-emptive garbage collection for ssd raid. Microprocessors and Microsystems, 56:13 – 21, 2018.

[33] N. R. Mielke, R. E. Frickey, I. Kalastirsky, M. Quan, D. Ustinov, and V. J. Vasudevan. Reliability of solid-state drives based on nand flash memory. Proceedings of the IEEE, 105(9):1725–1750, Sept 2017.

[34] A. Prodromakis, S. Korkotsides, and T. Antonakopoulos. MLC NAND flash memory: Aging effect and chip/channel emulation. Microprocessors and Microsystems, 39(8):1052 – 1062, 2015.

[35] A. M. Rahmani, M. H. Haghbayan, A. Miele, P. Liljeberg, A. Jantsch, and H. Tenhunen. Reliability-aware runtime power management for many-core systems in the dark silicon era. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(2):427–440, Feb 2017.

[36] S. S. Sapatnekar. What happens when circuits grow old: Aging issues in cmos design. In 2013 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA), pages 1–2, April 2013.

[37] B. Schroeder, R. Lagisetty, and A. Merchant. Flash reliability in production: The expected and the unexpected. In 14th USENIX Conference on File and Storage Technologies (FAST 16), pages 67–80, Santa Clara, CA, 2016. USENIX Association.

[38] B. Schroeder, A. Merchant, and R. Lagisetty. Reliability of nand-based ssds: What field studies tell us. Proceedings of the IEEE, 105(9):1751–1769, Sept 2017.

[39] J. Srinivasan, S. Adve, P. Bose, and J.A.Rivers. The case for lifetime reliability-aware microprocessors. In Int. Symp. Comp. Arch., pages 276–287, 2004.

[40] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The impact of technology scaling on lifetime reliability. In Int. Conf. on Dependable Systems and Networks, pages 177–186, June 2004.

[41] R. Subramani, H. Swapnil, N. Thakur, B. Radhakrishnan, and K. Puttaiah. Garbage collection algorithms for nand flash memory devices – an overview. In 2013 European Modelling Symposium, pages 81–86, Nov 2013.

[42] R. Verschoren and B. V. Houdt. On the impact of garbage collection on flash-based SSD endurance. In 4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads (INFLOW 16), Savannah, GA, 2016. USENIX Association.

[43] Y. Xiang, T. Chantem, R. Dick, X. Hu, and L. Shang. System-level reliability modeling for MPSoCs. In Conf. Hardware/Software Codesign and System Synthesis, pages 297–306, 2010.

[44] C. Zambelli, M. Indaco, M. Fabiano, S. D. Carlo, P. Prinetto, P. Olivo, and D. Bertozzi. A cross-layer approach for new reliability-performance trade-offs in mlc nand flash memories. In 2012 Design, Automation Test in Europe Conference Exhibition (DATE), pages 881–886, March 2012.