AgeSim: A Simulation Framework for Evaluating the Lifetime Reliability of Processor-Based SoCs

Lin Huang and Qiang Xu
CUhk REliable computing laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: {lhuang,qxu}@cse.cuhk.edu.hk

ABSTRACT

Aggressive technology scaling has an ever-increasing adverse impact on the lifetime reliability of microprocessors. This paper proposes a novel simulation framework for evaluating the lifetime reliability of processor-based system-on-a-chips (SoCs), namely AgeSim, which facilitates designers to make design decisions that affect SoCs' mean time to failure. Unlike existing work, AgeSim can simulate failure mechanisms with arbitrary lifetime distributions and do not require to trace the system's reliability-related factors over its entire lifetime, and hence is more efficient and accurate. Two case studies are conducted to show the flexibility and effectiveness of the proposed methodology.

1. INTRODUCTION

Today's system-on-a-chips (SoCs) typically contain one or more embedded processors. While the relentless scaling of CMOS technology has brought such processor-based systems with enhanced functionality and improved performance in every new generation, at the same time, the associated ever-increasing on-chip power and temperature densities make failure mechanisms such as electromigration and time dependent dielectric breakdown (TDDB) serious threats for the lifetime reliability of embedded processors [5, 24]. Industrial studies have shown that the failure rates for electrical systems within its warranty period can be very high and the main reason for such high failure rate was traced to over-heating of the embedded processors [17, 23].

Needless to say, designers need to make sure that their system meets the lifetime reliability requirement. To achieve this objective, there are many reliability-related decisions to make at design stage. For example, various dynamic power/thermal management (DPM/DTM) policies have been proposed for saving power and/or reducing power density in thermal hot spots and they have gained wide acceptance in the industry (e.g., [3, 6, 21]). These policies apparently affect processors' lifetime reliability significantly and we need to decide which policies to include in the design and how to tune their parameters under lifetime reliability constraint. In addition, providing fault-tolerance capabilities on-chip by incorporating redundant circuitries is an effective way for lifetime reliability enhancement [2, 25]. How much redundancy is enough to ensure the system's service life is an important decision to make at design stage to achieve reliable yet low-cost designs. Moreover, for multiprocessor SoCs (MPSoCs), how do we allocate applications to processors has a significant impact on the stress upon them and different allocation strategies may lead to remarkably different mean time to failure (MTTF) of the system [14]. Hence, again, when designers decide their task allocation strategies, they need to take the lifetime reliability factor into account.

Since the stress on processors vary significantly at runtime with different workloads, making the right decisions for the above mentioned design issues is extremely difficult, if not impossible, without an accurate lifetime reliability simulation framework. Obviously, it is unacceptable to build an experimental system and trace all the reliability-related factors over its lifetime (in the range of years) and use them for simulation. How to design an efficient yet accurate lifetime reliability simulator is therefore also a quite challenging problem, and there is only limited work in the literature in this domain [10, 19]. For the sake of simplicity, [10, 19] assumed an exponential lifetime distribution for each failure mechanism. In other words, the failure rate of the circuit is assumed to be only dependent on its instantaneous behavior (e.g., temperature and voltage), independent of its usage history. This assumption is apparently inaccurate: a typical wear-out failure mechanism will have increasing failure rate as the circuit ages even if the operational temperature and voltage remain the same [12, 13].

In this paper, we propose a novel aging-aware simulation framework for evaluating the lifetime reliability of processor-based SoCs, namely *AgeSim*. *AgeSim* can simulate failure mechanisms with arbitrary lifetime distributions and hence is able to take their aging effects into account, which results in more accurate simulation results. In addition, *AgeSim* does not require to trace the system's reliability-related factors over its entire lifetime. Instead, tracing the representative application flows running on embedded processors once is sufficient for our simulation without sacrificing its accuracy much. The main contributions of our work include:

- we propose a so-called *aging rate* concept to "hide" the impact of the SoC's reliability-related usage strategies (e.g., various DPM policies, trigger mechanisms, and application flow characteristics) with a single value, and we present a mathematical proof on how to express reliability function with aging rate. This novel concept enables us to simulate the representative workloads once instead of simulating the SoC's activities over its entire lifetime;
- we theoretically extend the above model to multi-processor systems with redundancy;
- we present a novel simulation flow that extracts the distributions of processors' activities when executing representative workloads, which facilitates us to obtain the system's performance, MTTF, and energy consumptions efficiently;

The remainder of this paper is organized as follows. In Section 2, we present preliminaries and motivation for this work. The proposed lifetime reliability simulation framework *AgeSim* is then introduced in Section 3. Next, Section 4 details the calculation of aging rate with the simulation results of representative workloads and validates its accuracy. We then extend the proposed lifetime reliability model for MPSoCs with redundant processor cores in Section 5. Two case studies are conducted in Section 6 to demonstrate the flexibility and effectiveness of the proposed methodology. Finally, Section 7 concludes this paper.

2. PRELIMINARIES AND MOTIVATION

There are many kinds of failure mechanisms that could result in permanent errors of integrated circuits (ICs). The most representative ones are electromigration (EM) on the interconnects, TDDB in the gate oxides, thermal cycling (TC), and negative bias temperature instability (NBTI) on PMOS transistors. These failure mechanisms have an increasingly adverse effect with technology scaling, and hence are serious concerns for the semiconductor industry.

978-3-9810801-6-2/DATE10 © 2010 EDAA

While the above failure mechanisms have been extensively studied at the circuit level historically, it is essential to investigate their impact at the system level when analyzing the lifetime reliability of processor-based systems. This is because, these failures are strongly related to the temperature and voltage applied to the circuit [1], while the processor's temperature vary significantly at runtime with different workloads. In addition, today's electrical systems are essentially adaptive systems, which change their runtime behaviors for power/thermal reduction. To be specific, the DPM and/or DTM policies being widely used in the industry include thermal throttling [9], module shutdown [4], dynamic voltage and frequency scaling (DVFS) [8], and task migration among processor cores [21]. All have significant impact on the stress upon the embedded processors and hence their failure rates, which makes the lifetime reliability analysis quite complex.

Srinivasan *et al.* [24] described a so-called RAMP model for lifetime reliability analysis for microprocessors and proposed to conduct dynamic reliability management (DRM) using this model. In this work, the authors assumed a uniform device density over the chip and an identical vulnerability of devices to failure mechanisms. Later, Shin *et al.* [20] introduced a structure-aware model that takes the vulnerability of basic structures of the microarchitecture (e.g., register files, latches and logic) to different failure mechanisms into account. Exponential distribution for failure mechanisms were assumed in [20, 24] (equivalently, there are *no* aging effects for failure mechanisms), which makes these models inherently inaccurate.

There were also some recent work on simulation-based lifetime reliability analysis, which can be used to evaluate different DPM policies [10, 19]. These simulators contain a power management unit, implementing DPM policies, and a reliability monitoring unit, which gathers reliability-related information in the system (e.g., temperature and voltage) and uses them to obtain instantaneous failure rates. Similar to [20, 24], failure mechanisms' aging effects were not considered and hence they lead to inaccurate simulation results.

With the more realistic non-exponential lifetime distributions, circuits' reliability at a specific time point t depends on both its current reliability-related factors (e.g., temperature) and its past aging effects. That is, even if a processor experiences the same stress at two different time points, their failure rates are different. To achieve accurate simulation, one possible method is to trace the processors' temperature and its execution parameters that affects reliability (e.g., voltage and frequency) throughout the entire lifetime, compute the corresponding lifetime reliability sequence, and finally integrate it over time t to obtain MTTF. Let us take the commonly-used Weibull distribution $\mathcal{R}(t) = e^{-(\frac{t}{\alpha})^{\beta}}$ for describing reliability function [1] as an example, where the scale parameter α depends on reliability-factors that changes at runtime (including temperature T and processors' execution state s) and shape parameter β hides the reliability-related factors that do not vary with time (e.g., structural properties of the circuits). Here, $\beta > 1$, if the failure rate increases over time. Depending on the temperature and execution state, the time horizon can be divided into a series of intervals (say, d intervals). By using this method, denoting by $\alpha(T_j, s_j)$ the scale parameters in the j^{th} interval and $\Delta_j\tau$ the interval length, the reliability at the d^{th} interval can be computed by $\mathcal{R}(t) = e^{-(\frac{\Delta_1\tau}{\alpha(T_1,s_1)} + \frac{\Delta_2\tau}{\alpha(T_2,s_2)} + \cdots + \frac{\Delta_{d-1}\tau}{\alpha(T_{d-1},s_{d-1})} + \frac{t-\sum_{i=1}^{d-1}\Delta_i\tau}{\alpha(T_d,s_d)})^{\beta}}$.

Recently, Karl *et al.* [15] considered general lifetime distribution for failure mechanisms, and proposed to conduct DVFS according to reliability budget. In this paper, to verify the effectiveness of their proposed DRM policy, the authors conducted a 10-year lifetime simulation in their experiments using the above method. They collected real workload data from desktop computers and fill the 10-year time with randomly selected 1-hour workloads. Within each 1-hour period, they used a single temperature value to calculate reliability. This ignorance of temperature variation within the period results in lack of accuracy for their simulation results. Using a fine-grained simulation can mitigate the accuracy problem, however, it would lead to unaffordable simulation time.

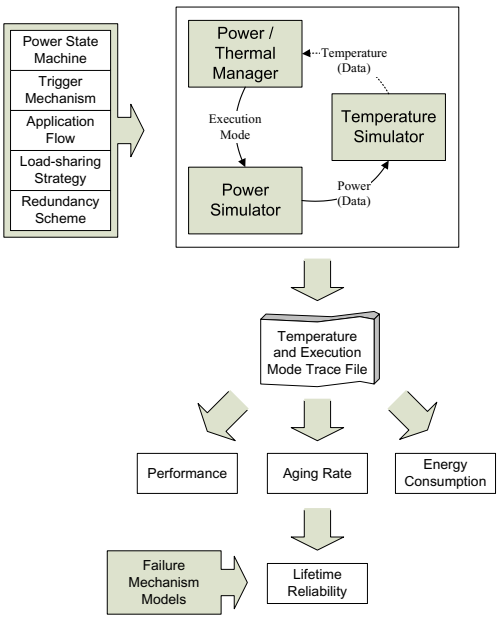


Figure 1: Lifetime Reliability Simulation Framework - *AgeSim*.

For systems at design stage, unlike in [15], it is impossible to obtain real workload information and simulate over its entire service life. In fact, due to the time-consuming temperature simulation, we can only simulate the system's execution for a short period. Therefore, we are facing the following challenging problem: *How to achieve efficient yet accurate lifetime reliability simulation with such limited information, when failure mechanisms follow arbitrary failure rate distributions?*

In addition, incorporating redundant circuitries on-chip is an effective way for lifetime reliability enhancement. Prior work (e.g., [10]) models multi-processor systems as parallel-serial systems [16] and calculates the lifetime reliability of the entire system accordingly. Using this model, however, also leads to inaccurate analytical results as it assumes all processors experience the same aging effects before they fail. Let us consider a standby redundant multi-processor system as an example. In such system, certain processors are initially set as spares and they become active only when some other active ones fail. Apparently, at the time point that a spare processor become active, it has a much smaller failure rate when compared to those processors that have already functioned for a long period. This effect, however, cannot be captured in the parallel-serial model. Consequently, *how to take the various aging effects of different processors in a multi-processor system when simulating its lifetime reliability is also a challenging problem.*

The above challenges motivate the proposed simulation framework investigated in this paper.

3. THE PROPOSED FRAMEWORK

Different from previous work, we propose to trace the representative workloads running on embedded processors in a fine-grained manner and use them to analyze the system's lifetime. This is feasible because as long as the probability of the target system being in each execution state and the temperature distribution obtained by the proposed approach conform to that in the whole service life, it can be used to represent the usage strategy of the system. In other words, the recorded information in this time duration is consistent with the usage strategy of the entire lifetime. Thus, if we can find out a quantity Ω (namely *aging rate*) that is able to capture the impact of the processor's usage strategy on its aging effect and at the same time it is *independent* of time t, we are able to evaluate the processors' reliability with arbitrary failure distribution at any time in its service life. Before describing how to calculate Ω in detail (see Section 4), let us present the overall lifetime reliability simulation framework in this section.

Our fine-grained simulator *AgeSim*, used to evaluate the influence of various usage strategies on processor-based SoCs, is composed of three closely-related parts: *power/thermal manager*, *power simulator*, and *temperature simulator*, forming a feedback control loop, as shown in Fig. 1. Here, *usage strategy* of a system includes its application flow characteristics (e.g., the distribution of application service time), power states, and trigger mechanism for state transitions. For systems containing more than one processor core, it also includes load-sharing strategy among multiple cores, and redundancy scheme (e.g., gracefully degrading system), if any. Note that, we mainly consider the lifetime reliability of processor cores in *AgeSim* as they typically experience the highest wearout stress in the system when compared to other hardware resources (e.g., peripherals). If, however, the reliability of these components are also of concern, our simulator can be easily extended to include them in the simulation framework.

The *power/thermal manager* determines the execution state of processors in the next time step based on what have occurred in the current time step. It is viewed as a black box, whose inputs and outputs are clear but can be implemented in any proper manner (power state machine is one of the choices [19]). It is worth noting that if the target system is a multi-processor system-on-a-chip (MPSoC), this part should include an application scheduler, which determines the processor cores that are used to execute each application. The *power simulator* evaluates the power consumption of every component according to their execution states and current application. The *temperature simulator* then takes the power consumption values and the temperature in the previous time step as inputs to obtain the temperature in the current time step. In *AgeSim*, we integrate HotSpot [21] into our simulator for accurate temperature computation. In our current implementation, temperature is used to trigger the execution state changes for a particular processor, if any. In case that the system's DPM/DTM policy requires other trigger mechanisms (e.g., processors' activity count [6]), they can be easily integrated into our simulation framework.

During simulation, we record the fine-grained temperature and execution state for every processor in each time step into a trace file. They are then used to compute the aging rate Ω and the expected lifetime of the system. At the same time, *AgeSim* also outputs the performance (e.g., mean response time) and energy consumption based on the traced information, which facilitates designers to evaluate their system from various aspects.

4. AGING RATE CALCULATION

According to earlier discussions, the key issue to achieve efficient yet accurate lifetime reliability simulation is to compute a time-independent aging rate Ω effectively with the limited traced information for representative workloads, so that we can express reliability as a function of Ω and t. This section shows how to achieve this objective using mathematical analysis. We tackle this problem by two steps: we first deduct a close-form lifetime reliability function with processors' time-varying operational states and temperature according to the reliability definition and property (Section 4.1), and then extract the time-independent aging rate parameter from this function (Section 4.2). The accuracy of the proposed model is validated in Section 4.3. It is important to note that, we target general failure distributions and we capture the aging impact of different workloads on processor cores, instead of simply averaging out the aging-related parameters.

4.1 Lifetime Reliability Calculation

Existing circuit-level reliability models for hard failure mechanisms are not readily applicable to analyze processors' lifetime because their operational state and temperature vary significantly at run-time. We therefore propose a new high-level analytical model in this work.

Let $\mathcal{R}(t,\Theta)$ be a general failure distribution, where Θ represents the general scale parameter by which time t is divided. For instance, α is the scale parameter in Weibull distribution $\mathcal{R} = e^{-(\frac{t}{\alpha})^{\beta}}$. As mentioned before, this parameter Θ is a function of temperature T. In addition, it also depends on several parameters that vary with processors' execution state, e.g., supply voltage and frequency for DVS-enabled processors. We therefore introduce another variable s to represent execution state and imply the state-related parameters. With these two variables, Θ can be written as $\theta(T,s)$. Without loss of generality, we assume there exists a set of possible execution states s and denote the set as $\mathcal{S}$.

As both T and s vary with respect to time t, we consider a finite sequence $0 = \tau_0 < \tau_1 < \cdots < \tau_d = t$ as a subdivision of time horizon $[0,t]$, including d intervals. Each interval j has the length $\Delta_j\tau = \tau_j - \tau_{j-1}$ and corresponds to a subsequence of temperature under a fixed operational state s_j. For all $\varepsilon > 0$, there exists $\delta > 0$ such that if the length of the largest interval $[\tau_i, \tau_{i+1}]$ ($i = \arg\max_j(\tau_{j+1} - \tau_j)$) is less than δ, the temperature variation within any interval is less than ε. Thus, for each interval j we can select an arbitrary temperature value from its temperature variation range for the entire interval, denoted as T_j. The corresponding scale parameter is therefore $\theta(T_j, s_j)$.

With the general failure distribution $\mathcal{R}(t,\Theta)$, since time t is divided by scale parameter, the reliability at time τ in the first interval (i.e., $\tau_0 \le \tau < \tau_1$) can be expressed as

$$R(\tau) = \mathcal{R}\big(\frac{\Theta}{\theta(T_1,s_1)} \cdot (\tau - \tau_0)\big), \qquad \tau_0 \le \tau < \tau_1 \tag{1}$$

Then, we move to consider the j^{th} interval, where $j > 1$. Denoting by c_j the accumulated aging effect at the end of the j^{th} interval. Apparently, $c_0 = 0$. The lifetime reliability at time τ ($\tau_{j-1} \le \tau < \tau_j$) is

$$R(\tau) = \mathcal{R}\big(c_{j-1} + \frac{\Theta}{\theta(T_j,s_j)} \cdot (\tau - \tau_{j-1})\big), \qquad \tau_{j-1} \le \tau < \tau_j \tag{2}$$

With this equation, at the end of this interval (i.e., $\tau = \tau_j^-$)

$$R(\tau_j^-) = \mathcal{R}\big(c_{j-1} + \frac{\Theta}{\theta(T_j,s_j)} \cdot (\tau_j - \tau_{j-1})\big) \tag{3}$$

This time point is also the beginning of the $(j+1)^{\text{st}}$ interval. Therefore, we have

$$R(\tau_j^+) = \mathcal{R}(c_j) \tag{4}$$

By the continuity of reliability function, we have $R(\tau_j^-) = R(\tau_j^+)$ and we can express c_j as:

$$c_j = c_{j-1} + \frac{\Theta}{\theta(T_j,s_j)} \cdot (\tau_j - \tau_{j-1}) \tag{5}$$

With this expression, we obtain the reliability at time t

$$\begin{aligned} R(t) &= \mathcal{R}(\Theta \cdot \sum_{j=1}^{d} \frac{1}{\theta(T_j,s_j)} \cdot (\tau_j - \tau_{j-1})) \\ &= \mathcal{R}(\Theta \cdot \sum_{j=1}^{d} \frac{1}{\theta(T_j,s_j)} \cdot \Delta_j\tau)) \end{aligned} \tag{6}$$

Taking limit as $\max_j \Delta_j\tau \to 0$ and $d \to \infty$, we have

$$R(t) = \mathcal{R}(\Theta \cdot \lim_{\substack{d\to\infty \\ \max_j \Delta_j\tau \to 0}} \sum_{j=1}^{d} \frac{1}{\theta(T_j,s_j)} \cdot \Delta_j\tau)) \tag{7}$$

In this equation, the state parameter s_j in any time interval j must belong to the set $\mathcal{S}$. We introduce an indicator function $\ ^{j}_{s}$ to represent whether the state in the j^{th} interval is s. That is to say, if in the j^{th} time interval the processor is in execution state s, this function equals 1; otherwise, it is zero. With this notation, we can express the aging effects in various states separately and rewrite Eq. (7) as their summation, i.e.,

$$R(t) = \mathcal{R}(\Theta \cdot \sum_{s\in\mathcal{S}} \lim_{\substack{d\to\infty \\ \max_j \Delta_j\tau \to 0}} \sum_{j=1}^{d} \frac{\ ^{j}_{s}}{\theta(T_j,s)} \cdot \Delta_j\tau)) \tag{8}$$

For integration, we define a filter function over time horizon $\ _s(\tau)$ such that it is one if τ falls into an interval with state s while zero otherwise. Therefore, we have

$$R(t) = \mathcal{R}(\Theta \cdot \sum_{s\in\mathcal{S}} \int_0^t \frac{\ _s(\tau)}{\theta(T,s)} \mathrm{d}\tau) \tag{9}$$

4.2 Aging Rate Extraction

In the above, we have successfully express processors' reliability function using their high-level operational states and temperature values. However, as we integrate $\frac{1}{\theta(T,s)}$ over time in Eq. (9), we have not obtained the time-independent quantity yet. Fortunately, on the time horizon, the temperature T is a function of time τ. With this observation, we define $\psi(T,s)\mathrm{d}T$ as the accumulated time in state s in an infinitesimal temperature interval $\mathrm{d}T$ around T and use it to substitute $\mathrm{d}\tau$ in Eq. (9), leading to

$$R(t) = \mathcal{R}(\Theta \cdot \sum_{s\in S} \int_0^\infty \frac{1}{\theta(T,s)} \cdot \psi(T,s)\mathrm{d}T) \tag{10}$$

The only term depending on time in Eq. (10) is $\psi(T,s)$. We use π_s to represent the probability a core being in state s, and $\nu(T,s)$ to indicate the conditional probability density function of a core having temperature T, given state s. These values can be easily extracted via simulation. Fig. 2 shows two example temperature distributions extracted from the trace file for applying random task allocation on a 9-core processor (see Section 6.2 for detail). The accumulated time can therefore be expressed as the product of three quantities, i.e., $\psi(T,s) = \pi_s \cdot \nu(T,s) \cdot t$. Substituting it back to Eq. (10), we have

$$R(t) = \mathcal{R}(\Theta \cdot \sum_{s\in S} \int_0^\infty \frac{1}{\theta(T,s)} \cdot \pi_s \cdot \nu(T,s) \cdot t\mathrm{d}T) \tag{11}$$

We define

$$\Omega \equiv \sum_{s\in S} \int_0^\infty \frac{1}{\theta(T,s)} \cdot \pi_s \cdot \nu(T,s)\mathrm{d}T \tag{12}$$

Since the current time t is independent of all terms in the definition of Ω, we have successfully obtained a time-independent variable Ω, referred as *aging rate*, to express the reliability at time t as

$$R(t) = \mathcal{R}(\Theta \cdot \Omega \cdot t) \tag{13}$$

It is necessary to highlight that, since the characteristics of the representative workloads is consistent with that of entire lifetime, this equation is applicable for any time t in the entire service life. Consider Weibull distribution, that is, $\mathcal{R}(t) = e^{-(\frac{t}{\Theta})^\beta}$. Substituting it into Eq. (13) yields

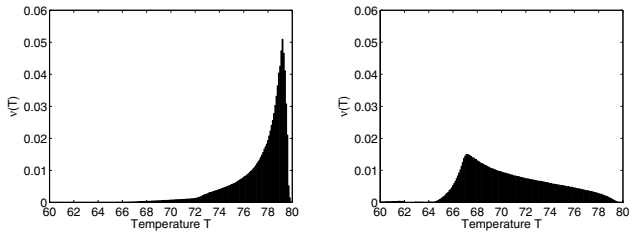
$$R(t) = e^{-(\frac{\Theta\cdot\Omega\cdot t}{\Theta})^\beta} = e^{-(\Omega\cdot t)^\beta} \tag{14}$$

Another point that should be highlighted is how to obtain the scale parameter $\theta(T_j,s_j)$ with its parameters T_j and s_j. As discussed before, existing circuit-level models for failure mechanism cannot be used directly, because they assume constant temperature and fixed aging-related parameters throughout the entire service life. For example, a widely-accepted lifetime reliability model for electromigration is given by $MTTF_{EM} \propto (V_{dd} \times f)^{-2} e^{\frac{E_a}{kT}}$ [11], assuming fixed absolute temperature T, supply voltage V_{dd} and clock frequency f. Here, E_a and k are material related constant and the Boltzmann's constant respectively. However, since temperature T_j and operational state s_j that implies state-related parameters (e.g., supply voltage and frequency) at the j^{th} time interval can be assumed as constant parameters with our fine-grained tracing, existing failure models can be used to calculate $\theta(T_j,s_j)$ for this particular time interval.

If the target system contains one or more processors without redundancy (the number of processor cores is n), given core i's aging rate $\Omega_{s,i}$ in each state s, the system's service life can be simply computed by integrating the system reliability over time t, i.e.,

$$MTTF = \int_0^\infty \prod_{i=1}^{n} \mathcal{R}\big(\Theta \cdot \sum_{s\in S} \pi_{s,i} \cdot \Omega_{s,i} \cdot t\big)\mathrm{d}t \tag{15}$$

Note that, while we mainly discuss MTTF in this work because it is one of the most important lifetime reliability metrics, with the extracted aging rate, *AgeSim* can be easily extended to analyze other reliability metrics (e.g., failure in time) and performability metrics (e.g., mean computation before failure).



(a) Run State (b) Idle State

Figure 2: Temperature Distribution Examples.

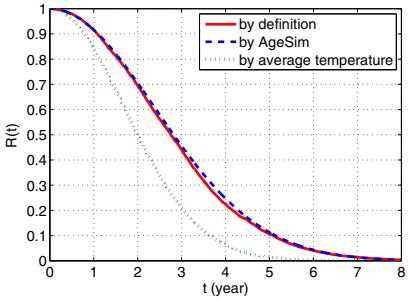


Figure 3: Accuracy Comparison.

4.3 Model Validation

Let us consider the 9-core processor again to demonstrate the accuracy of the proposed method. We compute its aging rate Ω with the trace file for one hour and then use Ω to calculate the system's lifetime reliability according to Eq. (13). For comparison, we have also tried to calculate the reliability according to the method used in [15]. Here, we trace the system operations for one hour and fill the system's service lifetime with workloads that are consistent to the system's usage strategy. We then use the average temperature value to calculate the lifetime reliability of the system. Both are compared with the results calculated by the definition of Weibull failure distribution directly, which has exactly the same workload as that for evaluating the method in [15]. The system lifetime reliability obtained using these three approaches are shown in Fig. 3. As can be observed from this figure, the proposed *AgeSim* can achieve almost identical reliability values when compared to that computed according to reliability definition. Using average temperature to obtain lifetime reliability, however, results in quite large errors.

It should be also noted that, the proposed method could be easily extended to analyze systems with multiple representative workloads (e.g., multi-mode MPSoCs [18]). We can organize these workloads into a hyper-workload according to their occurrence probabilities, and then take it as the input to *AgeSim*. Alternatively, we can extract the aging rate Ω_i and occurrence probability p_i for every execution mode with representative workload i. Similar to the above mathematical deduction, the unified aging rate is simply

$$\Omega_u = \sum_i \Omega_i \cdot p_i \tag{16}$$

5. LIFETIME RELIABILITY MODEL FOR MPSOCS WITH REDUNDANCY

In this section, we extend the previous model to analyze MPSoCs with redundant processor cores. Consider an MPSoC containing a set of $\{1,\cdots,n\}$ identical processor cores and it functions if no less than k components are good (e.g., Sony playstation game console requires seven out of eight synergistic processing elements in Cell processor to function [22]). The usage strategy of such a system can be changed a few times during its service life. For example, for gracefully degrading system, initially all cores are good ones and they share the system workload. Once a core fails, the workloads originally assigned to it need to be shared by the surviving cores, leading to heavier stress on them. Therefore, according to the usage strategy of cores, we divide the time horizon into several stages. In each stage, the usage of each core follows a fixed strategy.

Let us use $R_{i,\ell}(t)$ to denote the reliability of core i at stage ℓ. It depends on not only the characteristic at stage ℓ, but also that at previous stages. Without loss of generality, we assume core i can be in a series of states $\mathcal{S}_j$ at stage j. Accordingly, the aging rate and probability in state $s \in \mathcal{S}_j$ are referred as $\Omega_{s,i,j}$ and $\pi_{s,i,j}$, respectively. With these notations, the reliability of core i at stage ℓ can be expressed as

$$R_{i,\ell}(t) = \mathcal{R}\Big(\Theta \cdot \Big[\sum_{j=0}^{\ell-1} \sum_{s\in\mathcal{S}_j} \pi_{s,i,j} \cdot \Omega_{s,i,j} \cdot (t_{j+1} - t_j) + \sum_{a\in\mathcal{S}_\ell} \pi_{s,i,\ell} \cdot \Omega_{s,i,\ell} \cdot (t - t_\ell)\Big]\Big) \quad (17)$$

It is important to note that probably not all cores are surviving or in power-on state at this stage. Let $\mathcal{L}_\ell$ be the set of power-on surviving cores at stage ℓ. If any one of them fails, the system will leave stage ℓ and enter stage $(\ell+1)$ or becomes faulty. Therefore, the conditional probability that the system remains in stage ℓ at time t provided the past events $\mathbf{h}_\ell$ is given by

$$P_\ell^{sys}(t|\mathbf{h}_\ell) = \prod_{i\in\mathcal{L}_\ell} R_{i,\ell}(t) \quad (18)$$

The history $\mathbf{h}_\ell$ can be characterized by two vectors: the events $\mathbf{e}_\ell = \{e_1, \cdots, e_\ell\}$ and their occurrence time $\mathbf{t}_\ell = \{t_1, \cdots, t_\ell\}$. For instance, $\mathbf{e}_2 = \{\text{core 15 fails, core 6 fails}\}$ and $\mathbf{t}_2 = \{t_1, t_2\}$ represent that at time t_1 core 15 fails and at time t_2 core 6 fails. The vector $\mathbf{e}_\ell$ directly affects the set of good cores with power supply $\mathcal{L}_\ell$. We can therefore rewrite Eq. (18) as

$$P_\ell^{sys}(t|\mathbf{t}_\ell;\mathbf{e}_\ell) = \prod_{i\in\mathcal{L}(\mathbf{e}_\ell)} R_{i,\ell}(t) \quad (19)$$

To uncondition it, we consider vectors $\mathbf{e}_\ell$ and $\mathbf{t}_\ell$ separately. We do not know the exact occurrence time of the past ℓ failures, but we are certain that the ℓ^{th} event must occur before current time t (i.e., within time $[0,t]$), the $(\ell-1)^{\text{st}}$ event occurs before the ℓ^{th} one (i.e., in $[0,t_\ell]$). Hence, we have an inequality that $0 < t_1 < t_2 < \cdots < t_{\ell-1} < t_\ell$. On the other hand, since all power-on cores are likely to have failures, the possible first ℓ events of the system may not be unique. To include all possible cases, we denote them by a set $\mathcal{E}_\ell$. By the theorem of total probability, the unconditioned probability is

$$P_\ell^{sys}(t) = \int_0^t \mathrm{d}t_\ell \int_0^{t_\ell} \mathrm{d}t_{\ell-1} \cdots \int_0^{t_2} \mathrm{d}t_1 \sum_{\mathbf{e}_\ell\in\mathcal{E}_\ell} P_\ell^{sys}(t|\mathbf{t}_\ell;\mathbf{e}_\ell)\Pr[\mathbf{t}_\ell;\mathbf{e}_\ell] \quad (20)$$

Since our redundant system functions if no less than k cores are good, the reliability of such a system is equivalent to the sum of probability for a series of events that exactly ℓ core failures happen before time t. Its service life hence can be calculated as

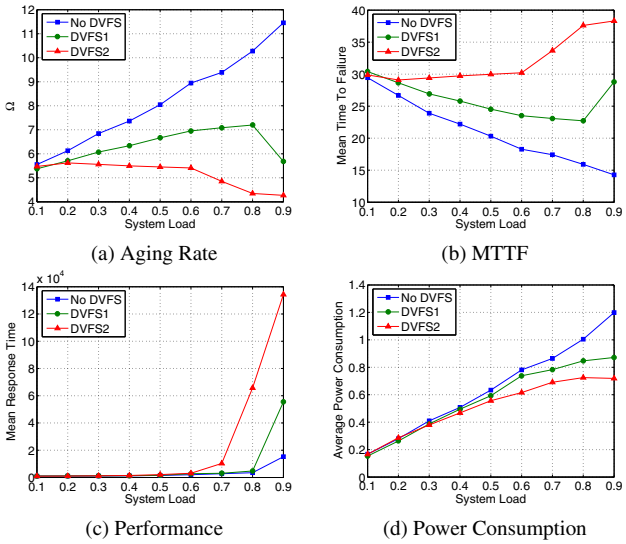
$$MTTF^{sys} = \int_0^\infty \sum_{\ell=0}^{n-k} P_\ell^{sys}(t)\mathrm{d}t \quad (21)$$

6. CASE STUDIES

In this section, we conduct two case studies to show the flexibility and effectiveness of the proposed *AgeSim* simulation framework. Due to the lack of public benchmark workloads, we use high-level synthetic workloads in our simulation, which are an application flow consisting of a large number of applications. For each application, their power consumptions at each execution state are given. We evaluate the lifetime reliability of processor-based SoCs with various system load ρ and it is obtained as follows. The application arrival to the system is assumed as a Poisson process with arrival rate λ, while the service time is an exponential distribution with mean $1/\mu$ in our case studies[1]. Denoting by n the number of cores in the system, system load ρ is defined as $\lambda/n\mu$. In practice, designers should provide the above information when running representative workloads on their systems.

Our framework is applicable for any failure mechanism model or the combination of these models. Due to the lack of public data on the relative weights for various failure mechanisms, however, we select to use electromigration model presented in [11] in our case studies and the parameters are set as follows: the cross-sectional area of conductor $A_c = 6.4\times10^{-8}cm^2$, the current density $J = 1.5\times10^6 A/cm^2$ and the activation energy $E_a = 0.48eV$. In addition, we use Weibull distribution to describe wearout effects with shape parameter $\beta = 4.0$, implying increasing failure rate with respect to time.

[1] Our framework is applicable for any application flow characteristics.



(a) Aging Rate (b) MTTF

(c) Performance (d) Power Consumption

Figure 4: The Impact of Dynamic Voltage Frequency Scaling.

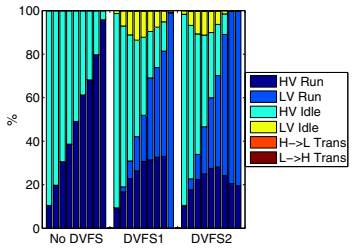
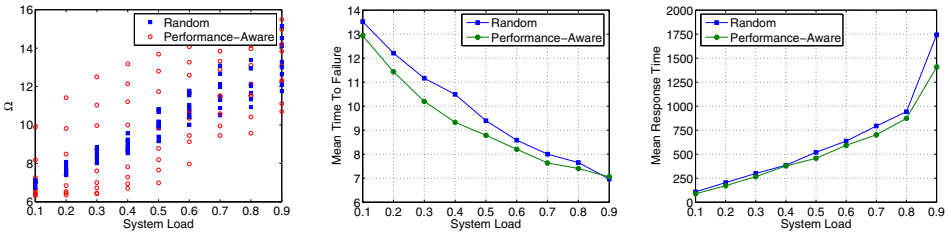


Figure 5: Accumulated Time in Each State.

6.1 Dynamic Voltage and Frequency Scaling

With DVFS, a processor core can be in one of four states: high voltage run, low voltage run, high voltage idle, and low voltage idle. Here, high voltage suggests the supply voltage V_{dd}, while low voltage corresponds to $90\%V_{dd}$ or $80\%V_{dd}$ (denoted as DVFS1 and DVFS2, respectively). A core is in run state if it has applications to perform, while in idle state otherwise. When the processor's temperature is higher/lower than a threshold T_H/T_L, it decreases/increases its supply voltage and frequency. The processor's voltage does not change for the transitions between run and idle state due to the associated high overhead. We set $T_H = 348.15K$ and $T_L = 338.15K$. Based on the model proposed in [7], the time required for voltage changes in DVFS1 and DVFS2 is $22\mu s$ and $44\mu s$, respectively. The frequency and power consumption in various states are computed according to the model presented in [4].

Fig. 5 shows the lifetime reliability metrics in these three cases with different system workloads and their expected service lives are shown in Fig. 5. When the system load is only 0.1, the aging rates caused by three configurations are almost the same. This is because, when the workload is too light, the DVFS policy is applied only in very rare cases. That is, the core alternates between high voltage run and high voltage idle states in the major portion of its lifetime, and seldom enters the low voltage states even if DVFS is used (see Fig. 5). Note that, the bars in each group in Fig. 5 (e.g., nine bars on the left side for *No DVFS* case) represents system load 0.1-0.9 from left to right. With DVFS1, when the system load increases, as long as the workload is not too heavy, the aging rate increases, but the growth rate is lower than that without DVFS due to the fact that the processor spends more and more time in low voltage states



(a) Aging Rate of Cores (b) Mean Time to Failure (c) Performance

Figure 6: Comparison of Task Allocation Schemes.

with the increase of workloads. An interesting phenomenon is that when the system load increases to 0.9, the aging rate of DVFS1 case decreases. We attribute it to the fact that the processor remains in the low voltage states all the time with such high workload (see Fig. 5). Next, we move to consider DVFS2, wherein frequency, voltage, and power consumption in low voltage states are much smaller than those with DVFS1. In this case, when the system load is greater than 0.2, its aging rate starts to decrease. This is due to the fact that the circuit spend more time in low voltage state with the increase of workloads, while the wearout stress in low voltage run state becomes even lower than that in high voltage idle state with DVFS2.

As discussed in Section 3, the performance and power consumption can also be obtained with *AgeSim*. The results for all cases are shown in Fig. 5 and Fig. 5, respectively. We observe severe performance degradation and some power savings by applying DVFS policy when the system load is high (e.g., $\rho \geq 0.8$).

6.2 Task Allocation on Multi-Core Processors

Due to process variation, processor cores in a homogeneous multi-core processor may have different operational frequencies. When an application arrives at the processor, a simple method is to randomly choose any available core to process this task, namely *random strategy*. To optimize performance, however, one may use the available core with the highest frequency to process it. We call this strategy as *performance-aware strategy*. Two application schedulers are implemented in our simulator accordingly.

Considering a 9-core processor with cores running at different frequencies (the maximum difference is 30%), we analyze the aging rate of each core with *AgeSim* and show the result in Fig. 6. When performance-aware strategy is used, we can observe significant variance among aging rates of different cores, especially when system workload is low. This is expected because those high-frequency cores are used much more often than those low-frequency ones under such circumstances. In contrast, the difference of cores' aging rates is relatively small if random allocation is assumed. Their impact on MTTF can be seen in Fig. 6, in which the lifetime reduction caused by unbalanced usage is quite serious when the system load is smaller than 0.5. When the system workloads become very high (e.g., $\rho = 0.9$), the aging rates with the two different allocation strategies are similar. This is because all processor cores are busy in most of their lifetime no matter which allocation strategy is chosen. In addition, the benefit of the performance-aware allocation strategy in terms of performance is shown in Fig. 6, which has a shorter mean response time.

7. CONCLUSION

With the relentless scaling of CMOS technology, the lifetime reliability of processor-based SoCs has become a serious concern for the industry. To meet the reliability requirement, designers need to know the impact of various usage strategies on the system lifetime. To facilitate this process, this paper proposes an accurate yet efficient simulation framework, which is applicable for evaluating any DPM/DTM policies, application flow characteristics and even task allocation algorithms, by tracing the system's reliability-related factors for representative workloads only. Two case studies are conducted to demonstrate the effectiveness of the proposed framework.

8. ACKNOWLEDGEMENTS

This work was supported in part by the General Research Fund CUHK417406, CUHK417807, and CUHK418708 from Hong Kong SAR Research Grants Council (RGC), in part by National Science Foundation of China (NSFC) under grant No. 60876029, and in part by a grant N_CUHK417/08 from the NSFC/RGC Joint Research Scheme.

9. REFERENCES

[1] Failure mechanisms and models for semiconductor devices (jep122c). *JEDEC Publication*, 2003.

[2] T. M. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In *Proc. MICRO*, pages 196–207, 1999.

[3] L. Benini, A. Bogliolo, and G. de Micheli. A Survey of Design Techniques for System-Level Dynamic Power Management. *IEEE Trans. on VLSI Systems*, 8(3):299–316, June 2000.

[4] L. Benini and G. D. Micheli. *Dynamic Power Management: Design Techniques and CAD Tools*. Springer-Verlag, 1997.

[5] S. Borkar. Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. *IEEE Micro*, 25(6):10–16, Nov.-Dec. 2005.

[6] D. Brooks and M. Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. In *Proc. HPCA*, pages 171–182, 2001.

[7] T. D. Burd and R. W. Brodersen. Design Issues for Dynamic Voltage Scaling. In *Proc. ISLPED*, pages 9–14, 2000.

[8] T. D. Burd, T. Pering, A Stratakos, and R. W. Brodersen. A Dynamic Voltage Scaled Microprocessor System. *IEEE Journal of Solid-State Circuits*, 35(11):1571–1580, Nov. 2000.

[9] J. Clabes et al. Design and Implementation of the POWER5 Microprocessor. In *Proc. ISSCC*, pages 56–57, 2004.

[10] A. Coskun et al. Analysis and Optimization of MPSOC Reliability. *Journal of Low Power Electronics*, 15(2):159–172, February 2006.

[11] A. K. Goel. In *High-speed VLSI Interconnections*. IEEE Press, 2nd edition, 2007.

[12] L. Huang and Q. Xu. On Modeling the Lifetime Reliability of Homogeneous Manycore Systems. In *Proc. PRDC*, pages 87–94, 2008.

[13] L. Huang and Q. Xu. Lifetime Reliability for Load-Sharing Redundant Systems with Arbitrary Failure Distributions. IEEE Trans. on Reliability, to appear.

[14] L. Huang, F. Yuan, and Q. Xu. Lifetime Reliability-Aware Task Allocation and Scheduling for MPSoC Platforms. In *Proc. DATE*, pages 51–56, 2009.

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

[16] I. Koren and C. M. Krishna. *Fault-Tolerant Systems*. Morgan Kaufmann, 2007.

[17] NVIDIA Provides Second Quarter Fiscal 2009 Business Update. http://www.nvidia.com/object/io_1215037160521.html.

[18] H. Oh and S. Ha. Hardware-Software Cosynthesis of Multi-Mode Multi-Task Embedded Systems with Real-Time Constraints. In *Proc. CODES+ISSS*, pages 133–138, 2002.

[19] T. S. Rosing, K. Mihic, and G. D. Micheli. Power and Reliability Management of SoCs. *IEEE Trans. on VLSI Systems*, 15(4):391–403, April 2007.

[20] J. Shin et al. A Framework for Architecture-Level Lifetime Reliability Modeling. In *Proc. DSN*, pages 534–53, 2007.

[21] K. Skadron et al. Temperature-Aware Microarchitecture. In *Proc. ISCA*, pages 2–13, 2003.

[22] Sony Computer Entertainment Inc. Cell Broadband Engine. http://cell.scei.co.jp/.

[23] SquareTrade. Report on Xbox 360 Failure Rates. http://blog.squaretrade.com/2008/02/xbox-fail-rates.html.

[24] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The Case for Lifetime Reliability-Aware Microprocessors. In *Proc. ISCA*, pages 276–287, 2004.

[25] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting Structural Duplications for Lifetime Reliability Enhancement. In *Proc. ISCA*, pages 520–531, 2005.