Lifetime Estimation for Core-Failure Resilient Multi-Core Processors

Wasala, S.M.; Niknam, S.; Pathania, A.; Grelck, C.; Pimentel, A.D.

DOI
10.1109/MCSoc60832.2023.00050

Publication date
2023

Document Version
Final published version

Published in
MCSoC 2023

License
Article 25fa Dutch Copyright Act (https://www.openaccess.nl/en/in-the-netherlands/you-share-we-take-care)

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 426, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)
Lifetime Estimation for Core-Failure Resilient Multi-Core Processors

Sudam M. Wasala*, Sobhan Niknam*, Anuj Pathania*, Clemens Grelck†*, and Andy D. Pimentel*

*Informatics Institute, University of Amsterdam, The Netherlands
†Institute for Informatics, Friedrich Schiller University, Jena, Germany

Abstract—Multi-core processors come with several cores integrated on a single die. They often work incessantly under high thermal stress, leading to severe wear-out. Server-class multi-cores already come with a mechanism to survive a core failure called Core Failure Resilience (CFR). Embedded multi-cores with CFR are already on the horizon. The surviving cores must take on an additional workload from their fellow failed core(s) under CFR. They must also operate on higher frequencies to continue meeting the target performance. However, this additional workload assignment further accelerates the wear-out of the surviving cores due to additional heat from higher frequency operation. Lifetime estimation frameworks rely on detailed simulations, which leads to long simulation times. These frameworks are unsuitable for the early stages of the design process as they cannot quickly evaluate many design points. Existing frameworks cannot estimate the Mean Time to Failure (MTTF) for multi-cores that include Core-Failure Resilient (CFR) capabilities. We introduce SLICER, the first framework for estimating the MTTF of CFR multi-cores. SLICER integrates with state-of-the-art tools HotSniper and MatEx for fast and accurate MTTF estimation.

Index Terms—Systems Simulation, Integrated Circuit Reliability, Lifetime Estimation

I. INTRODUCTION

Technology scaling enables multi-core processors to integrate several processing CPU cores on the same die. The integration provides a severalfold improvement in parallel processing capabilities in multi-cores. Multi-cores now form the core of general-purpose high-performance computing in data centers. However, this integration also subjects multi-cores to higher power densities than before. With higher multi-core power densities come higher multi-core operating temperatures. Multi-cores usually operate at much higher temperatures in data centers than at lower temperatures in data centers to satisfy the required user-imposed Service-Level Agreements (SLAs).

The high temperatures wear-out the cores within the multi-core by physically degrading its transistors. Thereupon, core failures are common in multi-cores deployed in data centers. Furthermore, scheduling decisions and workload variations can result in non-uniform core aging, resulting in cores failing in a multi-core at different times. State-of-the-art server-class multi-cores cater to the possibility of unequal core lifetimes with a feature called Core Failure Resilience (CFR) [1]. When a core fails, CFR allows the other cores of a multi-core to continue operations unaffected. CFR extends the lifetime of a multi-core, but it also comes with a heavy design and area overhead. Previously, embedded multi-cores rarely experienced a core failure as they operate intermittently at low temperatures. Consequently, the CFR feature is currently non-existent in embedded multi-cores. However, with the proliferation of embedded multi-core servers [2] and high-performance adaptive embedded systems [3], CFR capabilities are also expected in embedded multi-cores.

CFR provides the hardware mechanisms for multi-cores to continue operating with the remaining surviving cores even after several failures. However, the overlying CFR multi-core scheduler must reschedule the same workload on fewer cores to honor the SLAs. Therefore, the scheduler may operate the surviving cores at a higher frequency (for higher performance) using Dynamic Voltage and Frequency Scaling (DVFS) than before (provided that other constraints allow it to do so) as they take on the work of the failed cores. Surviving cores produce more heat operating at higher frequencies (and voltages). Consequently, DVFS may allow the multi-core to remain operationally adequate at the cost of accelerated core aging.

Estimating the lifetime for multi-cores is an active subject of research. We define Mean Time to Failure (MTTF) for CFR multi-cores as the estimated average time wherein the CFR multi-core (with the help of rescheduling) can still meet the SLAs with its surviving cores. State-of-the-art simulation frameworks such as LifeSim [4] allow the estimation of Mean Time to Failure (MTTF) for non-CFR multi-cores. However, no reliability simulation framework can estimate MTTF for CFR multi-cores out of the box. Estimating the reliability of multi-cores is quintessential in their design process. We present SLICER (Simulator for Lifetime estimation of Core-failure Resilient multi-cores), capable of estimating the MTTF of CFR multi-cores. MTTF for a multi-core is inherently subject to the underlying micro-architecture, (re)scheduling algorithm, reliability model, and system-level SLAs. SLICER provides a plug-and-play interface to provide all these necessary inputs. Designers can use SLICER to perform early design-space exploration for CFR multi-cores with respect to design parameters such as type and number of cores, (re)scheduling algorithms, DVFS policies, etc. while verifying that they meet the necessary reliability requirements. Additionally, SLICER can estimate the MTTF for a CFR multi-core
Monte collects (and potentially, SLICER Failure) interval produces temperatures ages the CFR multi-core based on the schedule. We present the first framework, HotSniper and reuses extracted traces HotSniper ELATED uses on-demand task-level trace extraction supports for the SLICER system simulators like Sniper to {\( n \)} or interval system simulators like SLICER. It is not time-wise feasible to estimate the MTTF using detailed low-level cycle-accurate [5] or interval simulations [6]. SLICER, therefore, uses isolated power-performance traces from the HotSniper [7] (and potentially Comet [8]) interval simulation toolchain for estimating MTTF. HotSniper provides traces for a given multi-threaded workload and microarchitecture at different core frequencies. SLICER provides full integration with HotSniper to automate the entire process of power-performance profiling. SLICER produces temperatures from the power traces (and user-provided floorplan) using the MatEx [9] thermal modeling tool. Finally, though configurable, SLICER uses the ElectroMigration (EM) [10] reliability model to estimate the MTTF from the thermals in this work.

Figure 1 shows the process flow for SLICER. It models applications as Directed Acyclic Graphs (DAGs) and supports DAG-based scheduling [11]. A heuristic scheduler maps the application DAG on the underlying CFR multi-core under a deadline (SLA) constraint to create a DVFS-based schedule. SLICER ages the CFR multi-core based on the schedule, repeating iteratively till the multi-core experiences a core failure. The scheduler then remaps the DAG on the remaining functional cores under the same deadline constraint. The scheduler reports success if it can find a viable schedule, and the process repeats till the next failure. It will indicate failure if no schedule can meet the deadline with the remaining cores. The failure provides a single Time-To-Failure (TTF) data point for MTTF estimation. SLICER uses randomized Monte Carlo simulations (with aging-dependent randomized core failures) to obtain the CFR multi-core MTTF.

Our Contribution: We present the first framework, SLICER, to estimate MTTF for CFR multi-cores. The framework provides a plug-and-play interface to insert the necessary input – micro-architecture, floorplan, DAG-scheduler, SLAs (deadlines), and reliability model. SLICER collects the necessary traces from the low-level HotSniper interval simulation toolchain and processes them with MatEx for the time-wise feasible estimation of MTTF.

Open Source Contribution: The source code for SLICER is released under the MIT license for unrestricted use and is available for download at https://github.com/sudam41/SLICER.

II. RELATED WORK

There are several frameworks available in the literature that estimate the lifetime of multi-cores [4], [12]–[17] (See Table I). However, most of them do not apply to CFR multi-cores. The frameworks in [13] and [16] assume a single core failure will fail the entire multi-core, resulting in an inaccurate MTTF for CFR multi-cores that can survive core failures.

Some frameworks in the literature estimate the lifetime of CFR multi-cores. However, they operate under the purview of several important limitations. These frameworks fall into two main groups, namely frameworks that use high-level power-performance simulations [12], [15], [18], [19] and frameworks that use low-level power-performance simulations [4], [13], [14], [16], [17]. A framework must be time-wise feasible to make it practical for performing early design-space exploration. Existing frameworks that use low-level simulations integrate with live cycle-accurate system simulators like gem5 [5] or interval system simulators like Sniper [6] to estimate MTTF. Integration with a live performance simulator brings accuracy but at the cost of heavy simulation time overhead in every run. Contrarily, frameworks that use a high-level approach define the workload of each task as an execution of an entire task. Such a high level of abstraction puts into question the real-world translation of the results. In contrast, SLICER uses on-demand task-level trace extraction from the integrated HotSniper and reuses extracted traces to minimize the invocation of low-level detailed simulators for lower simulation time overhead. This hybrid approach enables SLICER to reduce the simulation time while producing accurate results.

Furthermore, the frameworks in [4], [12]–[15] only consider multi-cores where all cores are identical. SLICER supports lifetime estimation for homogeneous and heterogeneous CFR multi-cores out-of-the-box thanks to its tight integration with HotSniper, which also supports heterogeneity. The framework described in [15] considers a scenario where it redistributes
workload after each core failure based on a predetermined configuration. This approach provides a solution to ensure schedulability in the hyperperiod where a core failure and task reallocation occur. However, it is unsuitable with dynamic reallocation scenarios where aging/thermal aware reallocations can be conducted [20]. The framework in [17] uses failure dependency graphs with sub-core units to estimate the lifetime of a CFR multicore. SLICER opts for core-level simulations as a trade-off to achieve faster simulations since SLICER targets early design-space exploration.

In addition to its application in this study, SLICER was employed internally within our research group for a distinct study in design space exploration to evaluate platform architectures and their floorplans [21].

### III. LIFETIME RELIABILITY MODELING OF A CORE

The permanent hardware failures that result in core failures manifest from core aging. A core (transistors within) inherently wears out under active processing. This wear-out is primarily a function of the core’s experienced temperature [22], [23] but can also involve other advanced parameters like operating voltage [24]. Technology scaling allows multi-cores to integrate more cores on a single die. However, numerous smaller cores in close proximity increase the power density of the cores, accelerating their aging.

The literature describes multiple wear-out mechanisms [25]. Core (transistor) aging is an active research subject. Researchers introduce more sophisticated aging models every year. It is not the intention of this work to keep up with them. We designed SLICER to work with any reliability model with a parameterized implementation. However, we must use some reliability model to obtain results in this work. Therefore, we chose to employ the commonly used ElectroMigration (EM) [25] model in this work. The core failure probability under the EM model is a function of the core’s temperature history — the operating temperatures and the temperature durations. The following equation gives the MTTF for a core under the EM model.

\[
MTTF_{EM}(T) = \frac{A_0}{(J - J_{crit})^n} \cdot \exp\left(\frac{E_{a}}{kT}\right) \tag{1}
\]

where \(A_0\) is a process-dependent constant, \(J\) is the current density, \(J_{crit}\) is the critical current density for the EM effect, \(E_{a}\) is the activation energy for EM, \(k\) is the Boltzmann’s constant, \(n\) is the material-dependent constant, and \(T\) is the operating temperature. \(J\) must be greater than \(J_{crit}\) for core failure under the EM model. We cannot use Equation (1) directly for system-level modeling. The core failures do not occur deterministically in practice. Equation (1) provides the expected value for core failure but not the probability distribution of failures around the mean. We use the common practice of employing Weibull (or log-normal) distributions to model the temporal core failure probability. Temporal core failure is the probability of a core surviving until a particular time. The following equation describes the reliability of a core with the Weibull distribution.

\[
R(t, T) = e^{-\left(\frac{t}{\alpha(T)}\right)^\beta} \tag{2}
\]

where \(t\) is the current time (measured in hours), \(T\) is the steady state temperature, \(\beta\) is the Weibull slope parameter, and \(\alpha(T)\) is the scale parameter that depends on the wear-out mechanism. The following equation defines the \(\alpha(T)\) for the EM model [15].

\[
\alpha(T) = \frac{A_0 (J - J_{crit})^{-n} \exp\left(\frac{E_{a}}{kT}\right)}{\Gamma(1 + \frac{1}{\beta})} \tag{3}
\]

where \(\Gamma()\) is the gamma function. Equations (2) and (3) give the reliability of a core at any given time. However, we require the entire temperature history of a core to calculate its MTTF. It is computationally expensive to calculate the temperature history. Instead, we calculate the average aging rate for a time interval and use this parameter as a proxy for temperature history to estimate the MTTF. The approximation does not introduce a calculation error by assuming that the workload repeats periodically. The following equation gives the average aging rate for a constant workload [15].

\[
\alpha = \frac{\sum_{i=0}^{p} \tau_i \cdot \alpha(T_i)}{\sum_{i=0}^{p} \tau_i} \tag{4}
\]

where \(\alpha\) is the average aging rate over \(p\) atomic steps. Each atomic step has a duration of \(\tau_i\) with temperature \(T_i\).

Core failure is inherently a random process. SLICER must decide which core to fail among all the surviving cores in a CFR multi-core. The probability of a core failure is negatively correlated with its aging, as given by Equation 2. However, the probability of any core failing is non-zero at any non-initial time. So, it can happen that a less-aged core fails first than a more-aged core by pure chance. Therefore, we use a random
function to determine the next core failure. We use an example to explain this mechanism next.

Figure 2 shows the fault distribution curves based on Equation 2 for two cores in a dual-core CFR multi-core. The multi-core has not experienced any core failure yet. Core 2 has aged more than Core 1, as hinted by its steeper fault distribution curve. SLICER chooses random floating point numbers $R_1$ and $R_2$ between 1 and 0 for Core 1 and 2, respectively. It then projects the random numbers temporally using fault distribution curves. Let $t_1$ and $t_2$ be the time corresponding to Core 1 and Core 2, wherein cores’ fault distribution curves attain the values of $R_1$ and $R_2$, respectively. In Figure 2a, since $t_2 < t_1$, the older Core 2 is the next failure. However, as shown in Figure 2b, if $t_1 < t_2$, the younger Core 1 is the next failure. The probability of the scenario in Figure 2a happening is higher than in Figure 2b, given that Core 1 is more aged. SLICER reallocates additional workload to surviving core(s) on a core failure. Furthermore, the cores probably run at higher frequencies under the new reallocated schedule than before to meet the deadline. Therefore, one can expect core temperatures for the surviving core(s) to change (most likely for the worse) after reallocation.

Figure 3 shows SLICER calculating a new fault distribution curve for each surviving core with the new temperature. Let $R'$ be the exact reliability value for a surviving core at the exact time of core failure $t'$. Let $t''$ be the hypothetical time wherein the surviving core would have achieved the reliability value of $R'$ when operating with the new – after reallocation – temperature from the beginning. Figure 3a shows the before- and after-reallocation fault distribution curve for a surviving core with $R'$, $t'$, and $t''$ projected on both curves. The new fault distribution curve must account for the depletion of reliability of the surviving core by the workload before reallocation. Therefore, the new fault distribution curve for the surviving core is a hybrid between the before- and after-reallocation curves. Figure 3b shows that the partial before- and after-reallocation fault distribution curves from Figure 3a form the first and second parts of the hybrid curve, respectively. The junction point wherein the two curves meet in time is given by the before reallocation fault distribution curve. Figure 3b shows the $R'$ and $t'$ projected on the hybrid curve.
IV. CFR MULTI-CORE MTTF ESTIMATION WITH SLICER

Figure 4 shows an overview of the SLICER framework. It takes in several inputs from the user: the hardware description (with CFR multi-core floorplan), the software application description (i.e., DAGs), the deadline (SLA), the (re-)scheduling policy, and the aging model. It produces the MTTF for the CFR multi-core as output. Below, we describe the process of obtaining the MTTFs from the inputs using various modules of the SLICER framework.

SLICER passes the user-provided hardware-software description to a module called Power-Performance Simulator. For this module, we use HotSniper [7] in SLICER. HotSniper simulates the software as compiled binaries, using detailed interval power-performance simulations. It provides core-level power traces at the finest supported granularity of 100 ns. Moreover, it also provides the start and end timestamps for each task in the software’s DAG. SLICER uses the timestamps to extract the power profile for each task from the temporal core-level power traces via the Profiler module. The Profiler passes the task-level power traces to the Scheduler module.

The Scheduler takes in the user-provided hardware-software description, scheduling policy, and the deadline (SLA) as input. It spatio-temporally maps tasks onto the functional (surviving) core(s) under DAG and deadline constraints with a user-provided heuristic scheduling policy. SLICER only supports static scheduling (other than a scenario where a core has failed). Therefore, the Scheduler assigns all tasks a fixed space in time in the schedule. SLICER does not support dynamic scheduling [26], as the workload needs to remain constant for the MTTF calculations to work. The Scheduler module considers the deadline hard and reports failure if the scheduling policy fails to find a schedule. It reports success and a feasible schedule otherwise. The Scheduler generates a new temporal core-level power trace corresponding to the feasible schedule and passes it to the Thermal Simulator module.

The Thermal Simulator generates a temporal core-level temperature trace corresponding to the power trace passed to it by the Scheduler. We employ the MatEx [9] thermal modeling tool inside the Thermal Simulator. SLICER passes the floorplan from within the user-provided hardware description to MatEx, which uses it to simulate CFR multi-core thermals. The Thermal Simulator passes the temperature trace to the Aging Simulator module.

The Aging Simulator models the wear-out of the processor. It takes in the user-provided aging model as input. It passes the temperature trace from the Thermal Simulator through the aging model iteratively to obtain the TTF for each core. It assumes the core with the smallest TTF to fail first. It passes which core exactly failed and the time from the last core failure to the Scheduler. The Scheduler then remaps the workload. The adaptive process repeats till the Scheduler fails to find a feasible schedule or all cores have failed. The Scheduler reports the MTTF to the user at the end of the process.

V. EXPERIMENTATION

Experimental Setup: We use the benchmark set STR2RTS [27] to evaluate SLICER. STR2RTS [27] comes with constructs that simplify task profiling of DAG-based StreamIt benchmarks [28]. We employ HotSniper [7] to generate power traces for each task in the benchmark. Some tasks in the benchmarks are tiny and do not produce power values with HotSniper, even at its smallest granularity of 100 ns. Therefore, we merged these tasks (per the DAG) to form larger tasks. Figure 5 shows the DAG transformation in the FMRadio benchmark due to task merging. Table II shows the number of tasks in each of the benchmarks. We simulate an Intel octa-core Nehalem-EX [1] as the underlying multi-core. Nehalem-EX is a high-performance CFR multi-core. We chose this system for experimentation due to the availability of its data and the CFR capabilities built into its architecture. Table III shows the corresponding hardware description. We set the ambient temperature to 45°C.

A. Impact of Core Failures

CFR multi-cores are unique in their ability to survive core failures. Failed cores do not directly impact the surviving cores. However, the workload reallocation that follows can affect the ongoing wear-out. The common understanding dictates that core failure accelerates wear-out for surviving cores due to additional workload. However, we observe that sometimes
### TABLE II: STR2RTS benchmarks used

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Number of tasks</th>
<th>Number of tasks after merging</th>
</tr>
</thead>
<tbody>
<tr>
<td>802.11a</td>
<td>117</td>
<td>73</td>
</tr>
<tr>
<td>Audiobeam</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>BeamFormer</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>CFAR</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CFIR</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>DCT</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>DES</td>
<td>423</td>
<td>337</td>
</tr>
<tr>
<td>FFT2</td>
<td>26</td>
<td>8</td>
</tr>
<tr>
<td>FFT4</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>FilterBank</td>
<td>53</td>
<td>10</td>
</tr>
<tr>
<td>FMRadio</td>
<td>67</td>
<td>42</td>
</tr>
</tbody>
</table>

### TABLE III: Hardware Description

<table>
<thead>
<tr>
<th></th>
<th>Number of Cores</th>
<th>ISA</th>
<th>Area of a core</th>
<th>L1 cache</th>
<th>L2 cache</th>
<th>L3 cache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
<td>x86</td>
<td>28mm²</td>
<td>32 KB</td>
<td>256 KB</td>
<td>24 MB</td>
</tr>
</tbody>
</table>

### TABLE IV: Simulation time for each benchmark

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Full Monte Carlo simulation time (s)</th>
<th>Average time per Monte Carlo iteration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>802.11a</td>
<td>648.89</td>
<td>3.24</td>
</tr>
<tr>
<td>Audiobeam</td>
<td>247.50</td>
<td>1.24</td>
</tr>
<tr>
<td>BeamFormer</td>
<td>294.56</td>
<td>1.47</td>
</tr>
<tr>
<td>CFAR</td>
<td>291.62</td>
<td>1.46</td>
</tr>
<tr>
<td>CFIR</td>
<td>262.66</td>
<td>1.31</td>
</tr>
<tr>
<td>DCT</td>
<td>292.99</td>
<td>1.46</td>
</tr>
<tr>
<td>DES</td>
<td>558.16</td>
<td>2.79</td>
</tr>
<tr>
<td>FFT2</td>
<td>303.82</td>
<td>1.52</td>
</tr>
<tr>
<td>FFT4</td>
<td>267.20</td>
<td>1.34</td>
</tr>
<tr>
<td>FilterBank</td>
<td>290.40</td>
<td>1.45</td>
</tr>
<tr>
<td>FMRadio</td>
<td>274.80</td>
<td>1.37</td>
</tr>
</tbody>
</table>

---

**B. Simulation Time of SLICER**

As SLICER is a tool for designers to perform early-stage design-space exploration, it provides fast predictions of the MTTF. Table IV provides the time spent by SLICER to determine the system’s MTTF when executing the different workloads from Table II. We used the HEFT algorithm [29] for task scheduling and conducted experiments on a Lenovo ThinkPad X1 laptop with an Intel Core i7 processor. Moreover, in this experiment, we used 200 Monte Carlo iterations for determining the MTTF, i.e., for each benchmark, we repeated the simulation – producing a TTF result – 200 times to determine the MTTF. The table also shows the average time for each Monte Carlo iteration (i.e., a single simulation instance). The simulation times captured here are for the main simulation loop only. We do not account for the time taken to generate and profile the power traces (see Figure 4), as this process only needs to be performed once for each workload-hardware combination. The simulation times show that SLICER only consumes a few seconds for an average Monte Carlo iteration, and a full Monte Carlo simulation takes an average of 5.6 minutes. A similar simulation would take a low-level simulator hours to days to complete. Thus, SLICER allows for efficient MTTF prediction, which enables early-stage design-space exploration where designers must evaluate many design points quickly.

**C. MTTF for Various Scheduling Algorithms**

SLICER provides the capability to analyze the impact of scheduling on the MTTF. Scheduling policies can play a key role in the lifetime of a CFR multi-core. Spatial mapping of a workload onto a core directly determines the core temperature, dictating its aging rate. Furthermore, the scheduler of a CFR multi-core processor has the added responsibility of reallocating the workload of a failed core amongst the surviving cores. We showcase the performance of three schedulers from the literature as a case study. We use HEFT proposed by Topcuoglu et al. [29] and DNDS/DWDS by Huang et al. [30] for the initial scheduling and the reallocation of tasks in our MTTF evaluation. The DWDS also performs DVFS when making scheduling decisions. Figure 6 shows the MTTF values for the three algorithms with different benchmarks. DWDS outperforms the other two schedulers for all benchmarks. The improvement in MTTF is especially prominent for benchmarks...
(a) $t = 0$ hours. (b) Core 1 failing at $t = 10634.3$ hours. (c) Core 6 failing at $t = 15066.0$ hours. (d) Core 4 failing at $t = 17466.6$ hours. (e) Core 8 failing at $t = 20191.1$ hours. (f) Core 2 failing at $t = 22437.5$ hours. (g) Core 7 failing at $t = 24897.5$ hours. (h) Core 3 failing at $t = 28882.2$ hours.

Fig. 7: Heat-map of a CFR multi-core surviving multiple core failures. "X" denotes a failed core. Note that this shows only a single instance of the Monte Carlo simulation and therefore the failing time is a stochastic TTF value and not the MTTF.

with many tasks, such as 802.11a and DES. We attribute the improvement over other schedulers to the efficient use of DVFS by DWDS to mitigate aging.

VI. VALIDATION

The lifetime of a microprocessor is in years. Such timescales pose an inherent obstacle in validating any form of a lifetime reliability simulator because any real-time validation experiment with real hardware would take years to complete. An alternative option would be to validate against another simulator, particularly one that operates at a low level. However, a full validation of MTTF estimation for CFR multi-cores with a lower-level simulator would also be time-wise infeasible.

Moreover, in attempting to find a suitable simulator to validate SLICER, we noticed that lifetime simulators that are operational and can be fairly compared to our work are difficult to find. Some simulator frameworks that are publicly available are at a very high abstraction level, such as simulators that represent workload in terms of percentages. This abstraction cannot be fairly compared to SLICER, which defines the workload as a DAG and uses power traces to simulate the MTTF. We found other simulators with obsolete code repositories that do not function as intended. As such, we have been unable to validate against another (lower-level) simulation platform.

Instead, we have addressed the validation of our simulation framework in a number of different ways. First, we have based our simulator on widely used tools and models, such as HotSniper and MatEx, which their authors have previously validated in isolation. Second, we have performed a substantial number of sanity checks that verified that the results produced by SLICER fall within reasonable and explainable ranges and exhibit explainable trends. Finally, one of the key novelties in our work is that we use isolated power traces for tasks. This design choice means that the accuracy of these traces does not account for the core interactions with other processor components, such as the other active cores and shared caches.

We investigated the impact of using the low-level thermal/power simulator HotSniper to generate isolated and parallel power traces and found that its effect is negligible. This observation confirms the validity of our choice for isolated power traces. We assume that a workload repeatedly and continuously executes (without idle times) until the system fails. While this may not resemble a real-life workload, we argue that this approach presents pessimistic (or at least conservative) results, which helps in lifetime predictions. Furthermore, such pessimistic MTTF predictions can still yield a high fidelity. For design-space exploration, the fidelity of MTTF predictions is typically more important than the absolute accuracy of the
predictions. With high fidelity, we mean that when comparing the MTTFs of multiple design candidates (to find an optimum design), the ranking of these candidates in terms of MTTF should be correct.

VII. CONCLUSION

Multi-core processors with CFR capabilities can remain operational after one or more core failures. CFR multi-core schedulers must reallocate the workload from the failed cores to the surviving cores while continuing to meet the performance, with the help of DVFS, if necessary. We present SLICER, a first-of-its-kind framework capable of predicting the MTTF of a CFR multi-core. SLICER allows the evaluation of different scheduling policies in the context of expected MTTF for CFR multi-cores. It also allows the user to define their own underlying hardware-software descriptions and reliability model beyond the ones provided by default. We demonstrate the capabilities of SLICER with the help of different scenarios and case studies. We plan to extend the framework to incorporate other features, such as the capability to simulate approximate computing and reliability-aware scheduling approaches that can further enhance the lifetime of systems with CFR multi-core processors.

REFERENCES