2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) | 979-8-3503-9361-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/MCSoC60832.2023.00050

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
Email: {s.m.wasala, s.niknam, a.pathania, a.d.pimentel}@uva.nl, clemens.grelck@uni-jena.de

***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 incessantly at high 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

This work has received funding from the European Union's *Horizon 2020* research and innovation program for the *APROPOS* project under the *Marie Skłodowska-Curie* grant agreement No. 95609 and the *ADMORPH* project under the grant agreement No. 871259.

2771-3075/23/$31.00 ©2023 European Union
DOI 10.1109/MCSoC60832.2023.00050

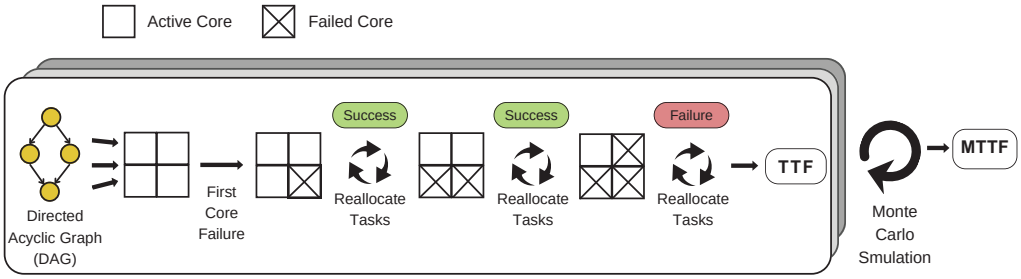


Fig. 1: Illustration of the process flow of *SLICER*.

for any predefined n number of core failures, adding further flexibility. As a consequence, by using $n = 1$, *SLICER* is also capable of simulating non-CFR multi-cores.

As *SLICER* is a tool for early-stage design-space exploration of multi-core systems, it operates at an abstraction level that allows fast simulations. For example, the framework models a core as a single unit and does not consider sub-core components. MTTF for multi-cores is in the order of years. 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 micro-architecture 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 a percentage or assign a single power value to represent the 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

Framework	High Level Simulation	Integration with Low-level System Simulation	Dynamic Task Reallocation	Heterogeneous Multi-cores
[15], [18]	✓	✗	✗	✗
[12], [19]	✓	✗	✗	✗
[16]	✗	✓	✗	✓
[4]	✗	✓	✓	✓
[13]	✗	✓	✗	✓
[14]	✗	✓	✗	✓
[17]	✗	✓	✓	✓
SLICER	✓	✓	✓	✓

TABLE I: State-of-the-art Simulation Frameworks

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.

$$MTTF_{EM}(T) = \frac{A_0}{(J - J_{\text{crit}})^n} \cdot \exp\frac{E_a}{kT} \tag{1}$$

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, β 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].

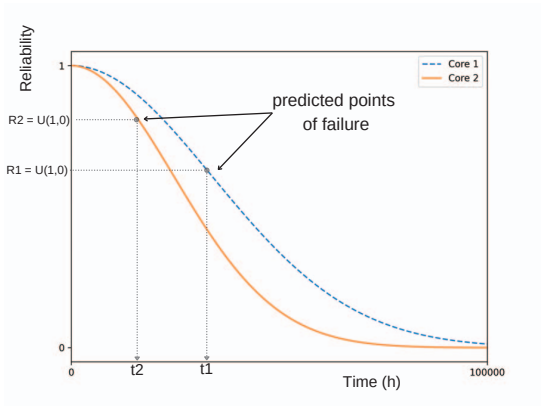
$$\alpha(T) = \frac{A_0(J - J_{crit})^{-n} \exp\frac{Ea}{kT}}{\Gamma(1 + \frac{1}{\beta})} \tag{3}$$

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].

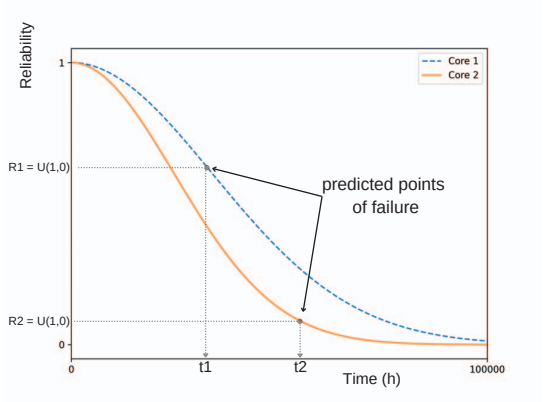
$$\alpha = \frac{\sum_{i=0}^{p} \tau_i}{\sum_{i=0}^{p} \frac{\tau_i}{\alpha(T_i)}} \tag{4}$$

where α is the average aging rate over p atomic steps. Each atomic step has a duration of τ_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

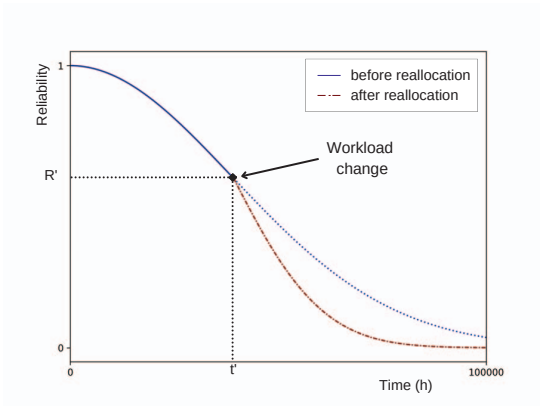


(a) Scenario 1: Older core fails first.

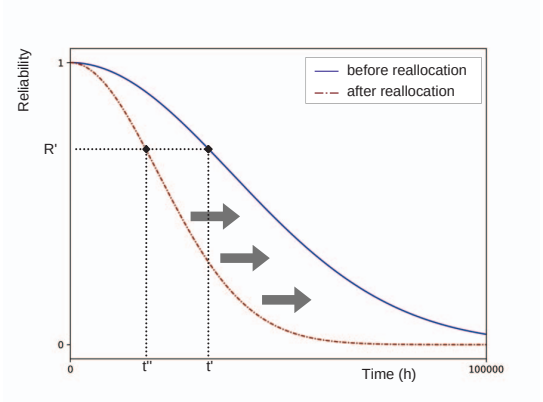


(b) Scenario 2: Younger core fails first.

Fig. 2: Mechanism for predicting the next core failure using fault distribution curves.



(a) Temperature-Driven Curve Shift



(b) New Hybrid Fault Distribution Curve

Fig. 3: The changes in fault distribution curve for a surviving core resulting from a workload reallocation after a failure.

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 $R1$ and $R2$ between 1 and 0 for Core 1 and 2, respectively. It then projects the random numbers temporally using fault distribution curves. Let $t1$ and $t2$ be the time corresponding to Core 1 and Core 2, wherein cores' fault distribution curves attain the values of $R1$ and $R2$, respectively. In Figure 2a, since $t2 < t1$, the older Core 2 is the next failure. However, as shown in Figure 2b, if $t1 < t2$, 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.

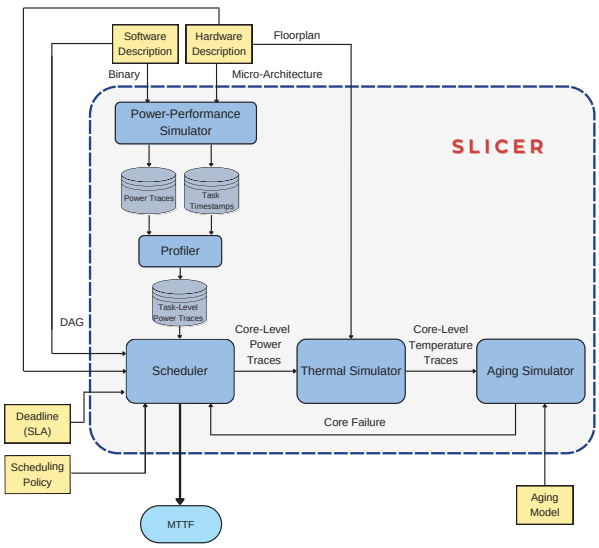


Fig. 4: An abstract overview of *SLICER*.

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.

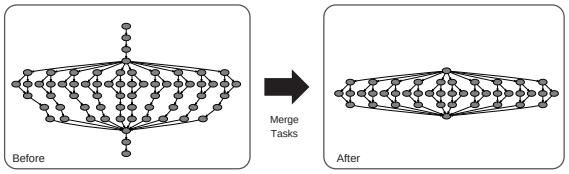


Fig. 5: Merging of tasks in *FMRadio* benchmark.

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

Benchmark	Number of tasks	Number of tasks after merging
802.11a	117	73
Audiobeam	20	20
BeamFormer	56	56
CFAR	4	4
CFIR	3	3
DCT	13	6
DES	423	337
FFT2	26	8
FFT4	10	7
FilterBank	53	10
FMRadio	67	42

TABLE III: Hardware Description

Number of Cores	8
ISA	x86
Area of a core	28mm^2
L1 cache	32 KB
L2 cache	256 KB
L3 cache	24 MB

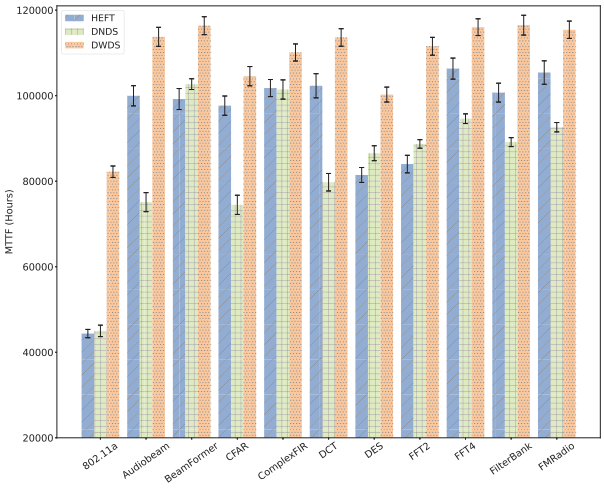


Fig. 6: Predicted MTTF with *SLICER* for different schedulers.

the wear-out of surviving cores slows down due to the failed cores acting as a heat sink for the surviving cores. *SLICER* is the first framework to provide such insights. Figure 7 shows the heat map of the *Nehalem-EX* processor running the *DES* benchmark (under a deadline constraint) at different stages of its lifetime. Figure 7(c) shows a scenario wherein the failed core (Core 1) acts as a heat sink for (Core 2), slowing down its wear-out. Consequently, Core 2 continues to operate at more or less the same temperature in Figure 7(b) as in Figure 7(c), even though the former has more workload than the latter. However, the effects do not last as the higher workload in Figure 7(d) temperature-wise overwhelms the Core 2.

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

TABLE IV: Simulation time for each benchmark

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 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

[1] S. Kottapalli and J. Baxter, "Nahalem-ex cpu architecture," in *2009 IEEE Hot Chips 21 Symposium (HCS)*. IEEE, 2009, pp. 1–19.

[2] P. Olivier, A. F. Mehrab, S. Lankes, M. L. Karaoui, R. Lyerly, and B. Ravindran, "Hexo: Offloading hpc compute-intensive workloads on low-cost, low-power embedded systems," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019, pp. 85–96.

[3] A. Pimentel, C. Grelck, L. Miedema, D. Sapra, M. Völp, F. Lucchetti, A. Matovic, M. Maggio, N. Vreman, S. Altmeyer *et al.*, "The admorph approach for adaptively morphing embedded systems," *Ada User Journal*, vol. 1, no. 1, 2023.

[4] R. Rohith, V. Rathore, V. Chaturvedi, A. K. Singh, S. Thambipillai, and S.-K. Lam, "Lifesim: A lifetime reliability simulator for manycore systems," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2018, pp. 375–381.

[5] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, aug 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718

[6] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[7] A. Pathania and J. Henkel, "Hotsniper: Sniper-based toolchain for many-core thermal simulations in open systems," *IEEE Embedded Systems Letters*, vol. 11, no. 2, pp. 54–57, 2018.

[8] L. Siddhu, R. Kedia, S. Pandey, M. Rapp, A. Pathania, J. Henkel, and P. R. Panda, "Comet: An integrated interval thermal simulation toolchain for 2d, 2.5 d, and 3d processor-memory systems," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 19, no. 3, pp. 1–25, 2022.

[9] S. Pagani, J.-J. Chen, M. Shafique, and J. Henkel, "Matex: Efficient transient and peak temperature computation for compact thermal models," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1515–1520.

[10] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," *ACM SIGARCH Computer Architecture News*, vol. 32, no. 2, p. 276, 2004.

[11] A. Maity, A. Pathania, and T. Mitra, "Pkmin: Peak power minimization for multi-threaded many-core applications," *Journal of Low Power Electronics and Applications*, vol. 10, no. 4, p. 31, 2020.

[12] L. Huang and Q. Xu, "Agesim: A simulation framework for evaluating the lifetime reliability of processor-based socs," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 51–56.

[13] A. Y. Yamamoto and C. Ababei, "Unified system level reliability evaluation methodology for multiprocessor systems-on-chip," in *International Green Computing Conference (IGCC)*. IEEE, 2012, pp. 1–6.

[14] S. Corbetta, D. Zoni, and W. Fornaciari, "A temperature and reliability oriented simulation framework for multi-core architectures," in *IEEE Computer Society Annual Symposium on VLSI*, 2012, pp. 51–56.

[15] C. Bolchini, M. Carminati, M. Gribaudo, and A. Miele, "A lightweight and open-source framework for the lifetime estimation of multicore systems," in *IEEE 32nd International Conference on Computer Design (ICCD)*. IEEE, 2014, pp. 166–172.

[16] N. Foutris, C. Kotselidis, and M. Luján, "Simulating wear-out effects of asymmetric multicores at the architecture level," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2019, pp. 1–6.

[17] A. Roelke, X. Guo, and M. Stan, "Oldspot: A pre-rtl model for fine-grained aging and lifetime optimization," in *IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 2018, pp. 148–151.

[18] C. Bolchini, L. Cassano, and A. Miele, "Lifetime-aware load distribution policies in multi-core systems: An in-depth analysis," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 804–809.

[19] L. Huang and Q. Xu, "Lifetime reliability for load-sharing redundant systems with arbitrary failure distributions," *IEEE Transactions on Reliability*, vol. 59, no. 2, pp. 319–330, 2010.

[20] Y. Shen, S. Niknam, A. Pathania, and A. D. Pimentel, "Thermal management for s-nuca many-cores via synchronous thread rotations," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.

[21] D. Sapra and A. D. Pimentel, "Exploring multi-core systems with lifetime reliability and power consumption trade-offs," in *Embedded Computer Systems: Architectures, Modeling, and Simulation: 23rd International Conference, SAMOS*, 2023.

[22] S. Niknam, A. Pathania, and A. D. Pimentel, "T-tsp: Transient-temperature based safe power budgeting in multi-/many-core processors," *Power*, vol. 4, no. 6, p. 8, 2021.

[23] S. Niknam, Y. Shen, A. Pathania, and A. D. Pimentel, "3d-ttp: Efficient transient temperature-aware power budgeting for 3d-stacked processor-memory systems." ISVLSI, 2023.

[24] I. Moghaddasi, A. Fouman, M. E. Salehi, and M. Kargahi, "Instruction-level nbti stress estimation and its application in runtime aging prediction for embedded processors," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1427–1437, 2018.

[25] J. S. S. T. Association *et al.*, "Failure mechanisms and models for semiconductor devices," *JEDEC Publication JEP122-B*, 2003.

[26] V. Venkataramani, A. Pathania, M. Shafique, T. Mitra, and J. Henkel, "Scalable dynamic task scheduling on adaptive many-core," in *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE, 2018, pp. 168–175.

[27] B. Rouxel and I. Puaut, "Str2rts: Refactored streamit benchmarks into statically analyzable parallel benchmarks for wcet estimation & real-time scheduling," in *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[28] W. Thies, M. Karczmarek, and S. Amarasinghe, "Streamit: A language for streaming applications," in *Compiler Construction: 11th International Conference, CC 2002 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8–12, 2002 Proceedings 11*. Springer, 2002, pp. 179–196.

[29] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[30] J. Huang, R. Li, J. An, H. Zeng, and W. Chang, "A dvfs-weakly dependent energy-efficient scheduling approach for deadline-constrained parallel applications on heterogeneous systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 12, pp. 2481–2494, 2021.