CHARM: A Checkpoint-based Resource Management Framework for Reliable Multicore Computing in the Dark Silicon Era

Venkata Yaswanth Raparti, Nishit Kapadia, Sudeep Pasricha
Department of Electrical and Computer Engineering
Colorado State University, Fort Collins, CO, U.S.A.
yaswanth@rams.colostate.edu, nkapadia@colostate.edu, sudeep@colostate.edu

***Abstract*—Transient faults due to single and multiple bit-flips are becoming increasingly common in today's multicore processing chips and reduce overall chip reliability. Simultaneously, aging effects due to phenomena such as Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI) also gradually reduce chip reliability over time. Unfortunately, with technology scaling, the increasingly stringent on-chip dark-silicon power constraints prohibit costly fault resilience solutions. Clearly, a viable approach is needed that can address both transient- and aging-induced faults in emerging multicore chips, but this is not easy. For example, techniques to slow down aging typically increase susceptibility to transient faults and vice-versa. In this paper, we propose a novel runtime framework (CHARM) to manage the useful chip lifetime, while also addressing transient faults and meeting performance and dark-silicon constraints. Experimental results on a 60-core chip multiprocessor show that CHARM not only achieves transient fault resilience while satisfying dark-silicon constraints, but also achieves an improvement of up to 2.5× in lifetime, and up to 6× in number of applications executed over the chip lifetime compared to a state-of-the-art solution.**

Keywords—Reliability aware mapping, aging, checkpointing, DVS

I. INTRODUCTION

With increasing transistor miniaturization, circuit densities have drastically increased, and the critical charge, which is the minimum charge capable of a bit-flip in a memory- or a logic-cell, has significantly decreased [1]-[2]. This phenomenon has caused newer process technologies to become more susceptible to transient-faults due to the effects of radiation, e.g., alpha-particle and neutron strikes. Simultaneously, circuit-aging due to phenomena such as Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI) is becoming prominent for systems manufactured at technology nodes of 45nm and below [3]-[4]. Such semiconductor-degradation causes gradual circuit slow-down over the operational lifetime of an electronic chip. The principal effect of such a circuit-aging mechanism is to increase circuit-threshold voltage (V_T), which results in higher circuit-delay. Such V_T-degradation causes slow-down of critical paths in cores and routers, thereby limiting overall system performance.

At the same time, the slowdown of power scaling with technology scaling, due to leakage and reliability concerns [5]-[6], has led to a rise in chip power-densities, giving rise to the dark-silicon phenomenon, where a significant fraction of the chip needs to be shut-down at any given time to satisfy the chip power-budget. With the extent of dark-silicon increasing every technology-generation (30-50% for 22nm) [7]-[8], chip multiprocessor (CMP) designs are becoming increasingly power-limited rather than area-limited. Runtime power-saving techniques such as dynamic voltage scaling (DVS) are thus becoming increasingly important. With asymmetric degradation of different cores on the CMP over the useful lifetime of the chip, and different applications requiring varying levels of minimum performance, dynamically scaling the voltage and frequency values of individual cores and intelligently mapping tasks to cores can yield significant benefits in terms of power, performance, and rate-of-aging trade-offs.

Additionally, recent works such as [9] have shown that by varying the degree of parallelism (DoP) of applications at runtime to adapt to the execution environment of the CMP, significant benefits can be achieved in terms of application service-times and power dissipation. Moreover, application-DoP (*app-DoP*) also impacts the application soft-error reliability, aging footprint, and chip power budget.

Figure 1 highlights the intricate inter-dependence between various optimization metrics (power, performance, reliability), design-knobs (voltage, app-DoP, task-to-core mapping) and their effects on physical phenomena (soft-errors, circuit-aging). Figures 1(a) and 1(b) show the plots of supply voltage (V_{dd}) versus average core power consumed, and the number of soft-errors observed (we model soft-errors using the approach in [16]), for three applications of varying memory intensities and fixed DoPs. In modern power-constrained designs with increasing levels of dark-silicon, operating at lower voltages, and hence lower frequencies, can reduce the average power consumption of the application as shown in figure 1(a). Given a fixed power budget, power savings translate to higher available power-slacks on the CMP-die thereby yielding higher performance by either simultaneously executing additional applications or running applications at higher app-DoPs. Additionally, low power techniques can potentially reduce the rate of aging on the die thereby extending useful lifetime of the chip. However, the soft-error rate (SER) exponentially increases when we reduce the rate of circuit-aging with DVS, as shown in figure 1(b). Higher app-DoP further increases the probability of soft-errors during the execution of an application. Even with sophisticated re-execution techniques using checkpointing and rollback, such as those proposed in [10], the occurrence of soft-errors can incur significant overheads in terms of power dissipation and aging footprint – the very metrics that were expected to improve by employing low-power DVS techniques.

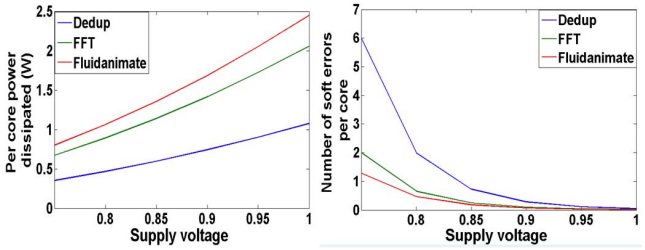

Figure 1. (a) Per-core power consumed by three applications of varying memory intensities at different supply voltages (V_{dd}); (b) Soft-errors observed per-core, when executing applications at different supply voltages

In this paper, we propose a novel system-level runtime soft-error and lifetime-reliability aware resource management framework (CHARM) that employs dynamically adaptable application degrees of parallelism (app-DoP), together with intelligent application mapping and DVS strategies to maximize the number of applications serviced over the target lifetime of a CMP, while meeting the chip-wide dark-silicon power budget (DS-PB) and application performance deadlines. For applications to recover from runtime soft-errors, we also integrate an error checkpointing and rollback scheme within our framework. Our novel contributions in this work are summarized as follows:

- we propose a novel runtime framework (CHARM) for application mapping and DVS that can adapt to different aging profiles of a chip and maximize the number of applications that meet their deadlines in the presence of soft-errors, over the chip lifetime;
- CHARM manages dynamically arriving applications by varying their application-DoPs as well as V_{dd} and execution frequency, based on *queue pressure* and *app-slack time*, to minimize checkpointing

and rollback overheads, and also to minimize the aging footprint;

- our methodology of evaluating maximum attainable performance, in the presence of soft errors and system aging, accounts for computing the execution time overhead due to checkpointing and rollback recovery, as well as V_T degradation over the lifetime of the chip;
- we also propose a novel aging aware network-on-chip (NoC) routing path allocation scheme that balances the core-router aging profile;
- our framework does not violate chip-wide power constraints while mapping an application, thus finding applicability in contemporary power-constrained (dark-silicon afflicted) CMP designs.

II. RELATED WORK

Several efforts (e.g., [11]-[13]) have proposed design-time fault-tolerant task scheduling frameworks that assume fault-detection mechanisms implemented on the multicore platform. Hardening techniques such as task re-execution and replication are utilized in these works to probabilistically meet task-completion deadlines for real-time tasks of varying criticalities, while minimizing energy. However, these efforts do not consider circuit-aging due to BTI, HCI stress, dark-silicon challenges, runtime adaptation support, or application DoPs.

In recent years, some researchers have proposed runtime and design-time techniques to address the problem of circuit-aging in CMPs. For example, Tiwari et al. [14] propose mapping high-power tasks onto faster (less-aged) cores and low-power tasks onto slower cores, thereby "hiding" the aging on the chip. They also propose to mitigate aging by scaling the supply voltage or the threshold voltage. Kapadia et al. [15] perform application mapping and DVS scheduling on the CMP while considering circuit-aging in cores and EM-aging in the power delivery network (PDN). Such an approach where tasks are mapped to cores without considering application-performance requirements would lead to higher number of applications being dropped and increase waiting time in the service queue. Also, these works neither perform soft-error management nor consider the benefits of adaptable app-DoPs.

A recent work [9] on runtime application scheduling for CMPs employs dynamically adaptable application DoPs to minimize average application service times and energy, while meeting a chip-wide power budget and soft-error-reliability constraints. However, the work ignores the circuit-aging phenomena and does not perform any lifetime optimization. Moreover, the work does not consider error recovery or task re-execution mechanisms for reliable system operation, and also does not emphasize satisfying completion deadlines of applications.

To overcome the abovementioned concerns, we propose a novel runtime resource management framework *(CHARM)* that, for the first time, considers the combined effects of runtime management techniques on lifetime-reliability and soft-error reliability of the CMP, to maximize the number of applications meeting their completion-deadlines, given a fixed CMP target-lifetime in years.

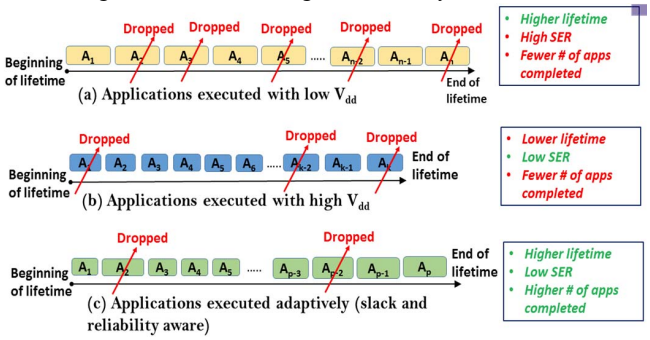

Figure 2: (a) application-scheduling representative of prior works [14], [15] where low V_{dd} is used to preserve chip lifetime; (b) application-scheduling where high V_{dd} is used to reduce SER probability; (c) application-scheduling where the V_{dd} and DoP are varied adaptively to reduce SER and maximize the number of completed applications within a target CMP lifetime.

III. MOTIVATION

In this section, we illustrate the advantages of our proposed *CHARM* framework with the help of a small example. We consider three scenarios in which applications arrive at runtime at a service-queue to be mapped onto the CMP. We call the chip as *aged* when the cores get slower due to degradation in threshold voltage V_T. An application can only tolerate up to a certain number of soft-errors beyond which the overheads of the recovery mechanism surpasses the available slack-time. When this happens, that application is *dropped* from the service queue without execution. Prior works [14]-[15] try to minimize aging footprint of applications by mapping them at low V_{dd} and using DVS scheduling to increase chip lifetime without effecting the performance, which is shown in figure 2(a). But they do not consider the effects of DVS and DoP on soft-error rate (SER). SER increases exponentially with the decrease in supply voltage-V_{dd}, as given in Eq. (5) in Section IV.A. Hence the chip encounters higher SER and higher number of dropped applications. Intuitively, scheduling applications with high V_{dd} as shown in figure 2(b) reduces SER along with the application execution times. However, this leads to much lower chip lifetime and considerably lower number of applications serviced over the lifetime.

In contrast, *CHARM* dynamically selects combinations of {DoP, voltage/frequency, mapping region} for the applications based on *app-slack time*, *queue pressure,* and *aging profile* to jointly minimize run times and maximize number of applications that meet their deadlines, as shown in figure 2(c). This leads to lower SER overhead and more applications meeting their deadlines within the power budget. As the chip gets *aged, CHARM* extends chip lifetime by mapping applications with lower voltage/frequency and higher DoP. This is done amidst a rise in SER probability, to scale down the power consumption and temperature footprint of the chip and also minimize aging of cores. The proposed approach with *CHARM* leads to more cores being active towards the end of the target chip lifetime, resulting in more number of applications meeting their deadlines over the lifetime.

IV. PROBLEM FORMULATION

A. Models for Performance and Reliability Estimation

We begin by modeling the maximum frequency of a core based on its supply voltage as given in Eq. (1).

$$f_{max} = \frac{\mu(V_{dd}-V_T)^{\alpha}}{C_0.V_{dd}} \quad \ldots\ldots (1)$$

where, V_{dd} and V_T are the supply voltage and effective threshold voltage of a core, α and μ are technology-dependent constants, and C_0 is switching capacitance of the critical path [16].

We model circuit-aging effects arising from BTI and HCI-induced circuit degradation, as these have been found to be the most dominant aging mechanisms in emerging semiconductor technologies.

Velamala et al. [18], [19] have shown that a Trapping/Detrapping (TD) based BTI model is capable of accurately predicting the degradation under a sequence of V_{dd}'s used in the DVS operation. They have noted that when the supply voltage is changed from a higher V_{dd} to lower V_{dd}, the circuit undergoes recovery. Therefore, our analysis of circuit-aging over the CMP-lifetime is based on the long-term aging prediction model proposed in [18], which accounts for different V_{dd}-levels over time. We estimate the effective ΔV_T increase that a component (core or NoC router) experiences over a time-interval of t using Eq. (2), where A, B and C are fitting parameters that are constant.

$$\Delta V_T(t) = L.[A + Blog(1 + Ct)] \quad .. (2)$$

$$L = K_1.\exp\left(\frac{-E_0}{kT}\right).\left\{exp\left(\frac{\beta V_1}{T_{ox}kT}\right).\alpha_1 + \cdots + exp\left(\frac{\beta V_S}{T_{ox}kT}\right).\alpha_S\right\} \quad .. (3)$$

In Eq. (3), V_i is the i^{th} V_{dd}-level utilized by the component for a time-duration α_i, S is the total number of allowable V_{dd}-levels, and $\{\alpha_1+\alpha_2+\ldots+\alpha_S\} \leq t$. T is the average temperature of the component during corresponding α_i. We use parameter-values in Eq. (2) and (3) from [18] and [19] that are validated against real silicon data.

HCI occurs because of the irreversible deposits of charges (holes/electrons) generated in regions such as the gate oxide, over the lifetime of the transistor. This phenomenon also leads to degradation of the threshold voltage (VT) which can be modeled as a function of stress-time [25]. The degradation model for VT from [26] is:

$$\Delta V_T = A.t^m \quad \text{..... (4)}$$

where, A and m are technology dependent parameters, and t is the time under which a component is under stress. We consider m to be equal to ~0.5 which is accepted over a wide range of processing technologies.

Transient faults are the bit flips in logic devices due to transient phenomena such as charged alpha particle strikes. We model the soft error-rates (SER) as discussed in [17]. We define $\lambda(f)$, as the SER at a given frequency f by the equation below:

$$\lambda(f) = \lambda_0.10^{d.(\frac{1-f}{1-f_{min}})} \quad \text{..... (5)}$$

where λ_0 is the SER corresponding to the highest frequency value (f_{max}). For compute cores we consider $\lambda_0 = 10^{-6}$ errors/sec and assume d=3 [9]. For NoC routers, we consider $\lambda_0 = 10^{-6}/3$ errors/sec as a router's area is approximately one third that of a core's area in the CMP platform we analyze in this work. We compute the probability of one or more faults occurring over an execution period τ using Eq. (6):

$$P(f,\tau) = 1 - e^{-\lambda(f).\tau} \quad \text{..... (6)}$$

$$E(f,\tau) = \int_0^{\tau} \tau.P(f,\tau).d\tau \quad \text{..... (7)}$$

Eq. (7) gives the expected number of faults $E(f,\tau)$ observed in a given time interval [0, τ] during which f remains constant. We compute the number of faults in any given interval [τ_1, τ_2] as follows:

$$E[\tau_1,\tau_2] = E(f,\tau_2) - E(f,\tau_1) \quad \text{..... (8)}$$

We employ a checkpoint and rollback based error recovery mechanism as proposed in [20]. The number of checkpoints employed for a task is a function of its worst case execution time L and its deadline D. Using this technique, fault-free execution time of an application involves two overheads: checkpoint time-(C_i) and time to sanity check the processor state-(S_i). Occurrence of a failure further incurs time to retrieve the latest saved state and re-execution time-(R_i). The following equations show the overheads as a function of number of checkpoints:

$$T_i(n) = T_i + n.C_i + (n+1).S_i \quad \text{..... (9)}$$

$$F_i(n) = T_i + R_i + k.\left(\frac{C_i}{n+1}\right) + S_i \quad \text{..... (10)}$$

Eq. (9) gives the fault-free execution time of a task i with n checkpoints ($T_i(n)$) where, T_i is the ideal execution time, $n.C_i$ represents checkpoint time and $(n+1).S_i$ represents sanity check time. Eq. (10) gives the execution time of a task i in the presence of k faults ($F_i(n)$) The term $k.(C_i/n+1)$ represents the re-execution time for k faults. For our system, we tolerate up to three faults, k=3, as we have observed that the overhead of tolerating more faults per task in applications we analyze surpasses the available slack-time in the best case. *CHARM* decides n based on the deadline D_i of the task graph to be mapped. Eq. (11) gives the optimum number of checkpoints n_i assigned to a task i:

$$n_i \leq 2.\frac{C_i}{D_i - T_i - R_i - S_i} - 1 \quad \text{..... (11)}$$

where, D_i is the deadline of the task i, T_i is the fault free execution time, R_i is the re-execution overhead and S_i is the sanity check overhead. The periodic checkpointing interval duration for each task i is thus T_i/n.

B. Inputs, assumptions and problem objective

We assume the following inputs to our problem:

- A 2D mesh NoC-based CMP of dimension $\{dim_x, dim_y, dim_z\}$ and number of tiles N=$dim_x \times dim_y \times dim_z$; each tile has a core and a router;
- A chip-wide dark-silicon power budget (DS-PB);
- An application task graph for each application; vertices with task execution-times on compute cores and edges with inter-task communication volumes; error-free execution times and volume values are assumed available from offline profiling;
- A set of candidate supply voltages $(V_{dd}) = \{V_1, V_2, \ldots V_n\}$ for each core; maximum frequency of a core for this V_{dd} is given by Eq. (1);
- Application task graphs for the set $P = \{P_1, P_2, \ldots P_n\}$ of DoPs for all applications; an application i possesses $|P_i|$ viable DoPs; an application has a maximum DoP value beyond which performance does not improve (or gets worse due to high synchronization overheads) - such sub-optimal DoP values are ignored;
- A set of permissible rectangular shapes of regions that an application can be mapped on to $\{B_1 \ldots B_n\}$; e.g., a set {2×4, 4×2} for an application with DoP=8.

We make the following assumptions in our work:

- There exists one-to-one mapping between tasks and cores, i.e., a core can execute only one task (thread) at any given time;
- Applications are mapped contiguously on non-overlapping rectangular shaped regions of the 2D CMP for inter-application isolation and optimized communication-profiles (as recommended in prior works such as [21]); thread-migration is not considered;
- Per-core granularity of DVS is considered, to meet DS-PB and application performance demands, and facilitate runtime selection of DoP for the applications to avoid execution deadline violations;
- Similar to prior works [14], [15] we assume the presence of on-chip sensors to detect the V_T degradation along the critical path circuits of cores and routers due to aging and send that information to our framework; we also assume the presence of an on-chip error detection mechanism to detect soft-error events in cores and routers and initiate the application re-execution process from a checkpoint;
- On-chip aging sensors monitor the runtime V_T values of individual cores and routers at the end of each *epoch* and send the values to our framework, to enable smart mapping decisions; an *epoch* is defined as a time-period during which the aging profile of the chip is assumed to be constant or does not grow significantly;

Objective: Given the above inputs and assumptions, the objective of our *CHARM* framework is to dynamically determine application-specific mapping (region selection, task-to-core mapping, and NoC routing), DoP values, and checkpoint periods, as well as a per-core DVS schedule, to maximize the number of applications that meet their execution-deadlines, while satisfying chip-wide DS-PB and tolerating up to k transient faults per application over a given chip target lifetime.

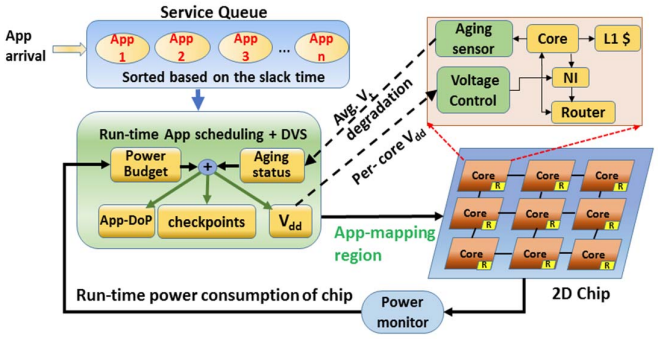

Figure 3: Overview of CHARM runtime app-DoP selection, reliability aware mapping, and DVS scheduling framework.

V. CHARM FRAMEWORK: OVERVIEW

The key aspects of our proposed framework are illustrated in figure 3. *CHARM* makes decisions based on the runtime input it receives from on-chip aging sensors and *app-slack time* = {worst case execution time – deadline} available for an application waiting in the queue. *CHARM* intelligently prioritizes between lifetime and performance based on the available app-slack time and the chip degradation profile. *CHARM* dynamically selects the DoP, checkpoint period, and per-core V_{dd}, for an application's execution, based on runtime inputs and available slack. *CHARM* operates as two nested loops, *(i)* circuit-aging, lifetime and epoch management (figure 4(a), outer loop); and *(ii)* reliability-aware application mapping, DVS, and application-DoP selection (figure 4(b), inner loop). These two components are discussed in detail next.

A. Circuit-aging, lifetime, and epoch management

As discussed earlier, the lifetime of a chip is divided into epochs. In each epoch, applications arrive to the CMP for execution. *CHARM* maps them immediately or keeps them in a service queue for mapping later. The applications waiting in the service queue are sorted at every occurrence of an *app-event* in the increasing order of their app-slack times. An app-event is defined as either the arrival of a new application to the CMP or the exit of a mapped application from the CMP. At an app-event, *CHARM* successively removes applications from the service queue and maps each one of them until there are insufficient number of consecutive idle cores on the chip to execute an application without violating the application deadline and the chip DS-PB constraints.

The inner loop performs reliability-aware mapping, app-DoP selection, and V_{dd} selection during an epoch are discussed in section V.B (shown as the pink box in figure 4(a), with an expanded view in figure 4(b)). The output of this inner loop at the end of the epoch provides information about the activity on the chip over the epoch, such as number of applications executed in the epoch, the active-times (AT's) of compute-cores and NoC routers over the epoch, and the thermal profile of these components over the epoch. Given these system-stats for the last epoch, the rise in effective V_T values (ΔV_T's) of all cores and NoC routers on the CMP (i.e., extent of BTI and HCI-induced circuit aging) is calculated as discussed in section IV.A using the V_{dd}'s and temperatures experienced by these components during all of their AT's over the entire epoch. The computed ΔV_T's are saved and passed on to the inner loop for reliability aware mapping, DoP and V_{dd} selection (section V.B) in the next epoch.

Note that at the start of the very first epoch, the V_T's are initialized with nominal values representing no degradation and the ΔV_T's are initialized to zero-values. When the end of lifetime condition is encountered, the framework stops mapping applications, and outputs the lifetime of the chip along with the total number of applications executed over the lifetime. *We consider the chip as failed (end of lifetime) when an application is dropped despite there being no other application running on the CMP and when the overall chip aging-profile (sum of tile V_T values) is beyond a certain threshold.*

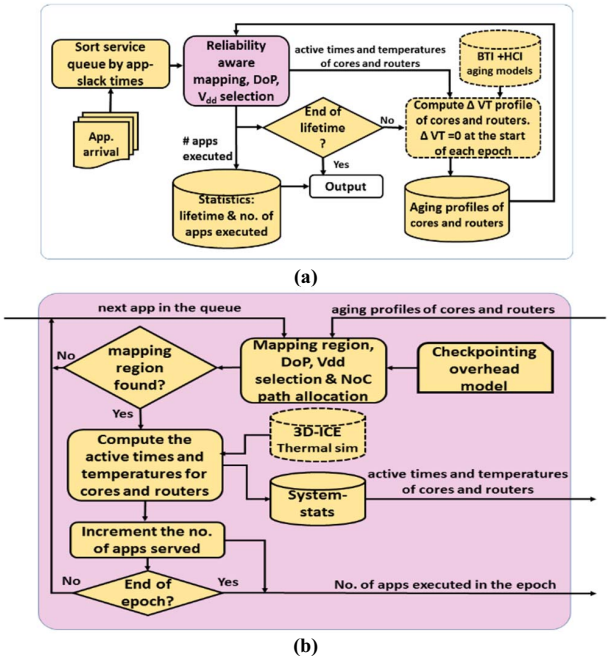

Figure 4: CHARM design-flow: (a) circuit-aging, lifetime and epoch management (outer-loop, Section V.A); (b) reliability aware mapping, DVS and app- DoP scheduling scheme (inner-loop, Section V.B); blocks shown with dotted outlines are simulated models used in our work, and are a proxy for on-chip sensors that will get the information at runtime in a real system.

B. Reliability aware mapping, DVS and app-DoP selection

Before mapping, *CHARM* assigns checkpointing periods for each application as given by Eq. (11). Subsequently, the worst case execution time of an application is estimated, given the overheads of checkpointing and rollback recovery mechanism, using Eq. (9), (10). If the worst-case execution time cannot meet the application-slack time using the available on-chip resources, that application is dropped from the service queue. For any application under consideration, this stage consists of three phases, *(i)* region selection, app-DoP and V_{dd} selection, *(ii)* communication-aware task-to-tile mapping, and *(iii)* NoC routing path allocation. We describe each of these steps below.

(i) region selection, app-DoP and voltage-selection: In our framework, an application with a given DoP can be mapped on to a rectangular region on the 2D CMP, with shapes to be chosen from a pre-defined list $\{B_1, \ldots, B_n\}$ for that application. All intra-application communication is contained within the region, thus application-isolation is maintained and communication cross-interference between applications is eliminated. Our heuristic in this step utilizes the runtime V_T-degradation profile of the CMP, which is given as:

$$\Omega = \sum_{k=1}^{k=N} V_{Tk}\,, \qquad \ldots\ldots\ldots (12)$$

where V_{Tk} is the average V_T value of the k^{th} tile (and includes core-V_T and router-V_T for the tile), and N is the total number of tiles on the CMP. The objective of our heuristic changes according to the value of Ω: *when the value is greater than a threshold ζ the objective is to preserve the lifetime of the CMP, otherwise the objective is to maximize the number of applications that complete before their deadlines.* When the objective is to preserve lifetime, we define a metric ψ to select the rectangular region on the CMP:

$$\psi = \sum_{k=1}^{k=DoP} \frac{\max\_V_T - V_{Tk}}{\max\_V_T - nom\_V_T} \qquad \ldots\ldots (13)$$

where, V_{Tk} is the same as in Eq. (12) for cores within the region; and nom_V_T is the nominal (lowest) effective V_T-value of a core with no aging. We define max_V_T as the maximum V_T value that the core can support (at highest V_{dd}). In order to preserve lifetime, CHARM selects a region with the least ψ that satisfies the application's deadline and DS-PB constraints. This is done because tiles dissipate higher leakage power when they are less aged, resulting in higher on-chip temperatures that cause further aging of the chip. Hence, it is intuitive to choose aged-cores that satisfy the application deadline, to preserve the overall lifetime of the chip.

Algorithm 1: Reliability-aware region, DoP and V_{dd}-selection heuristic

Inputs: V_T-profile, $\{P_1, \ldots P_\eta\}$, $\{B_1, \ldots, B_n\}$, $\{V_1, \ldots V_n\}$, DS-PB

```
1:  for each DoP in {P1,..Pη} & each Vdd in {V1,...Vn} & each tile in CMP, do{
2:    for each shape in {B1, ..., Bn} do {
3:      list _time.insert(Pi,Bi,Vi)
4:      list_age.insert(Pi,Bi,Vi)
5:    } // end for each shape
6:  }// end for
7:  if (Ω < ζ) and (app-slack time is less than τ) {
8:    ptr = list _time.begin( )
9:  } // end if
10: else if ((app-slack is greater than τ) or (Ω ≥ ζ) {
11:   ptr = list _age.begin( )
12: } // end if
13: while(ptr != list _time.end( )) do{
14:   check if CMP meets DS-PB with the ptr ->(Pi,Bi,Vi)
15:   if (DS-PB constraint not met ): ptr++
16: } // end while
17: if ( (Pi,Bi,Vi) is not found) drop the application
```

output: a valid region to map the application, app-DoP value and V_{dd}-level for cores in the selected region; or application being dropped

Algorithm 1 shows the pseudo code for our region, DoP and V_{dd} selection heuristic. The heuristic performs a search over the available per-core supply voltages ($V_1, \ldots V_n$), application DoPs $\{P_1, \ldots P_\eta\}$ and the permissible mapping regions $\{B_1, \ldots, B_n\}$ for the application. Aging profiles (V_T) of cores and routers, permissible DoPs P_i, permissible

shapes B_i and per-core voltages are given as inputs to Algorithm 1. With these inputs and the DS-PB constraint, our heuristic aims to find a suitable {DoP, mapping region and V_{dd}} *combination* for an application under consideration for mapping to the CMP.

For any candidate combination of a given application, the heuristic estimates the number of soft-errors, as per Eq. (8), and computes an *estimated executed time*, as per Eq. (10). If the *estimated execution time* for a combination cannot meet the application deadline, the combination is not considered for mapping. The mapping heuristic does an exhaustive search over combinations of all the DoP (P_i), V_{dd} (V_i) and mapping regions (B_i) and sorts them into two ordered lists (**lines 1-5** in Algorithm 1). The first list, *list_time*, is in the increasing order of the *estimated execution time*, for a (P_i,B_i,V_i) combination. The second list, *list_age*, is in the increasing order of the region's aging profile ψ, given by Eq. (13). If the total aging profile of the CMP (Ω from Eq. (12)), is less than a threshold ζ, and if the app-slack time is less than τ, the heuristic prioritizes faster app execution time and finds a suitable candidate from *list_time* (**lines 7-9**). In all other cases, it prioritizes lifetime preservation and finds a suitable candidate in the *list_age* (**lines 10-12**). The heuristic then starts at the beginning of the list, where the best combination is saved, and checks if that meets the DS-PB constraint. If not, it iterates to the next combination in the list (**lines 13-16**). If none of the combinations satisfy the DS-PB constraint, the application is dropped from the service queue (line 17).

When the application arrival rate is very high, our aim is to map more applications on the CMP. *Hence, when the queue pressure is above a threshold 'χ' (queue-pressure > χ), the application under consideration is mapped over a smaller DoP range,* as given by Eq. (14), (15):

$$Q.size > \chi : P = \{P_1, P_2, \ldots P_k\} \quad \ldots.. (14)$$

$$Q.size \leq \chi : P = \{P_1, P_2, \ldots P_N\} \quad \ldots.. (15)$$

where $Q.size$ gives the size of the service queue, and $\{P_1, P_2, \ldots P_k\}$ is a trimmed down list of permissible DoPs, with $P_k < P_N$.

We now present the theoretical time complexity of our heuristic. At most N tiles (total tiles on the 2D NoC-based CMP) are considered for the prospective mapping region. Note that the permissible DoPs $|D|$, V_{dd} levels $|S|$ for the cores and number of admissible shapes n are very small constant integers, and the DoP of the application (relatively small integer c – treated as a constant) number of tiles are to be evaluated at each of these iterations. With these assumptions, our region/*Vdd*/DoP-selection heuristic effectively runs in linear-time complexity with respect to the number of tiles, N: O($c.n.|D|.|S|.N$), and is very suitable for fast execution at runtime with low overhead.

(ii) Communication-aware task-to-tile mapping: After the mapping region for an application has been selected (of size equal to app-DoP), our heuristic proceeds to map the application's task-graph on to the CMP tiles. We use a fast and efficient task-to-tile incremental-mapping approach (similar to that used in prior works such as [22]) suitable for use at runtime, which aims to minimize communication between cores. We consider the earliest deadline first (EDF) task scheduling scheme for each application task graph and map that task-graph on to a selected rectangular shaped region of tiles on the 2D CMP;

(iii) Wear-out balancing routing path allocation (WBR): After the task to tile mapping step, we map the communication-flows of the current application on to the NoC in the selected region on the CMP. We propose an aging- and congestion-aware routing scheme (WBR) to balance the core-router aging profile and extend the lifetime of the NoC routers along with that of the cores. WBR trades-off aging with network-congestion in the NoC by selecting routing paths to maximize NoC-lifetime while leveraging the knowledge of maximum execution time constraints of the mapped application.

To ensure an implementation with low-overhead, path diversity, and deadlock freedom, our routing algorithm builds on the Odd-Even partially adaptive turn model routing scheme [24] for 2D-mesh NoCs. The Odd-Even scheme typically presents multiple routing options at each hop. The key idea of our routing algorithm is to select the next hop (and hence a path) in such a way that it either preserves lifetime of the routers or reduces congestion. We designed a cost-function for next-hop selection during routing that considers the difference between router-aging and core-aging (*router_*V_T *– core_*V_T) values to ensure balanced aging in CMP tiles. Moreover, as congestion in the NoC-links leads to excessive routing delays and thus longer application-runtimes, we also prefer allocating communication flows to links with lesser communication-volumes. The following routing cost function (Rt_{cost}), which is a linear combination of the two normalized metrics, is used to make routing decisions at each hop along the path:

$$Rt_{cost} = \alpha_R.\frac{(V_T\ difference)-(minimum\ V_T\ difference)}{range\ of\ V_T\ difference} + \beta_R.\frac{(volume)-(minimum\ volume)}{range\ of\ volume} \quad \ldots.. (16)$$

where, α_R and β_R are weighting coefficients, V_T *difference* represents (*router_*V_T *– core_*V_T) of the candidate next hop router, and *volume* represents the existing communication-volume (already allocated while routing previous flows) on the link. WBR selects the next hop with the minimum routing-cost, Rt_{cost}, given in Eq. (16). The values of the coefficients in Eq. (16), α_R and β_R, are re-evaluated after routing each flow, as shown below:

$$\beta_R = \{current\ app.\ delay\}/\{\delta.(app.\ execution\ time\ constraint)\}$$
$$\alpha_R = (1 - \beta_R) \quad \ldots.. (17)$$

Before any application communication flows are mapped through the NoC routers, we start with values α_R =1 and β_R =0. As flows are mapped and the estimated application-delay increases, the value of β_R increases (and α_R decreases) proportionally until the application-delay reaches a significant fraction (δ) of the application execution time constraint. At this point (β_R =1 and α_R =0), WBR ceases to be aging-aware and routes on paths with minimum congestion exclusively, to satisfy the execution time constraint of the given application. Algorithm 2 below summarizes our wear-out balancing NoC routing scheme.

Algorithm 2: *Wear-out balancing routing path allocation*

Inputs: Task-graph, execution time constraints, minimum frequency, task-mapping of current application, V_T-profile of compute-cores and routers

1: Initialize α_R=1 and β_R =0
2: for all communication-flows **do {**
3: for all hops on the minimal path **do {**
4: select the next hop with the least Rt_{cost} as per Eq.(16) **}**
5: update α_R and β_R as per Eq. (17)
6: }

output: all flows of the application allocated in the CMP-region

The paths that are allocated for each communication flow are stored in lightweight routing tables, in each NoC router. In this work, the largest size of an application footprint (DoP) is 32, which has an upper bound of 64 communication flows through any router. Hence, each router will have to store the next hop information for at most 64 paths (64 entries in a table). Assuming 2 bits for the output port and 6 bits for the source and destination each (for the 60-core CMP we consider), the footprint of the NoC routing table is only 768 bits. Thus, the hardware overheads of implementing WBR are low and reasonable.

VI. EXPERIMENTAL STUDIES

A. Simulation setup

We conducted experiments on 13 different parallel applications from the SPLASH-2 [27] and PARSEC [28] benchmark suites. These applications are executed on the cycle-accurate SNIPER [29] multicore simulator together with McPAT [30] to obtain their steady state power, compute time, and communication intensity across various app-DoPs, V_{dd} and clock frequencies. The DoP values we used ranged from 4 to 32 beyond which most of the applications were observed to have lower performance due to high communication (synchronization) overheads. The *CHARM* framework is implemented in C++ and is integrated with the SNIPER simulator, models for aging and soft-errors, and checkpointing and rollback support. *CHARM* also uses the open-source

thermal emulator 3D-ICE [23] to simulate the temperature profile of the cores and routers from the steady state power values, assuming a conventional air-cooled heat-sink. For the given power-profile, 3D-ICE outputs the core and router temperatures on the die.

We categorized the 13 benchmarks into two groups: *(i)* memory-intensive benchmarks - {*cholesky, fft, radix, raytrace, dedup, canneal, and vips*); and *(ii)* compute-intensive benchmarks – {*swaptions, fluidanimate, streamcluster, blackscholes, radix, bodytrack, and radiosity*}. As *radix* has properties of both, we use it in both groups. In our analyses, we employ three types of application sequence groups as inputs to our framework: memory-intensive, compute-intensive, and mixed (using all 13 applications). Each application-sequence has 100 randomly ordered application-instances selected from the respective group. To enhance the statistical significance of our results, we averaged results over four different randomly generated application-sequences for each group. To simulate the chip-lifetime within a reasonable time, we extrapolate the effects of aging over 500 such sequences, making the total number of application-instances executed within an epoch to be approximately l = 50,000. We experimented with three different inter-application arrival rates of 1.4s, 2.8s and 5s. We also analyzed the impact of different soft-error rates.

We considered a 2D CMP with 60 homogeneous tiles, fabricated at 22nm. Each tile has an x86 core, a NoC router, and a private L1 cache. The tiles are arranged in a 10×6 mesh layout. The V_{dd} values supported by each tile (core + router) are between 0.75V-1V, in steps of 0.05V. Note that as all the cores on which a parallel application is mapped are operated at the same V_{dd} and clock frequency, and as there is no inter-application communication, there are no overheads due to voltage level converters and clock synchronizers during application execution. We considered the computation frequencies using Eq. (1) and obtain the respective application execution time from the SNIPER simulation data. The dark-silicon power budget (DS-PB) is assumed to be 80W for our selected 60 core chip at the 22nm node. We also considered the runtime overhead for our framework. The total execution time for CHARM on one of the cores is 0.36ms, which is insignificant compared to the average application execution time (~seconds).

For computing the circuit-aging, we assumed the nominal V_T of each core and router to be 0.3V at the beginning of the chip lifetime. We consider a tile to be unusable after its average V_T goes beyond 0.57V. Above that value, the maximum operating frequency of the core cannot meet any of the applications' deadline constraints. The chip aging-footprint threshold ζ depends on workload types and arrival rates. The ζ value was empirically derived, as discussed in the next section.

In the WBR routing heuristic, we use δ=0.6, to calculate the value of β_R, for an appropriate trade-off between application performance and aging. In our experiments, an epoch interval can range between 25 to 35 days, depending on the power profile, execution-times, and average DoPs of the application workload, as well as the degree of aging in the chip. Also, as the overheads incurred due to employing aging sensors have been reported to be quite small (power dissipation of 84.7nW, sensing-latency of 100µs, and area of 77.3µm^2 per sensor at 45nm technology node [31]), we ignore them in our analysis.

B. Simulation Results

We compare our *CHARM* framework against an enhanced version of a prior work *VARSHA* [9] that tries to optimize the energy and performance of a multicore chip while meeting dark-silicon power constraints as well as satisfy reliability and performance constraints. The work however does not employ any checkpointing and rollback based correction, nor does it consider circuit-aging phenomenon while mapping. While *VARSHA* does not consider per-core DVS, we have enhanced *VARSHA* with per-core DVS in our analysis for a fair comparison with *CHARM* which also assumes per-core DVS support. We explore three variants of our *CHARM* framework: *CHARM-5*, which is designed for a target lifetime of 5 years; *CHARM-7*, which is designed for a target lifetime of 7 years; and *CHARM-NA*, which has no target lifetime. *CHARM-NA* thus only has the soft-error prevention mechanism, and aims for high V_{dd} and app-DoP to get the best execution speeds throughout the chip lifetime.

We simulated and analyzed the lifetime of the chip, total number of applications executed over the lifetime, average power dissipated by applications, and the number of functional cores at the end of the lifetime, for the four frameworks. Figure 5 shows the lifetimes of the CMP for the different frameworks. *CHARM-7* and *CHARM-5* are designed to achieve their target lifetimes of 7 and 5 years respectively. This is made possible by changing the threshold value ζ for different target lifetimes and application arrival rates. For *CHARM-7*, ζ is empirically derived to be approximately 21.5V for arrival-rate-1, 22.5V for arrival-rate-2, and 23V for arrival-rate-3. Similarly for *CHARM-5* it is approximately 27V for arrival-rate-1, 28.5V for arrival-rate-2, and 30V for arrival-rate-3. From figure 5(a), 5(b) and 5(c) it is evident that *CHARM-5* and *CHARM-7* either meet or come very close to achieving their target lifetimes of 5 years and 7 years respectively. Both *CHARM-7* and *CHARM-5* intelligently adapt their mapping phases to save lifetime or optimize performance according to the available slack time and threshold constraints. Without a target lifetime, *CHARM-NA* optimizes primarily for performance while *VARSHA* optimizes for energy, leading to their lower lifetimes. In particular, *CHARM-7* achieves 50-100% improvement in lifetime compared to *CHARM-NA*, and up to 2× improvement compared to *VARSHA*.

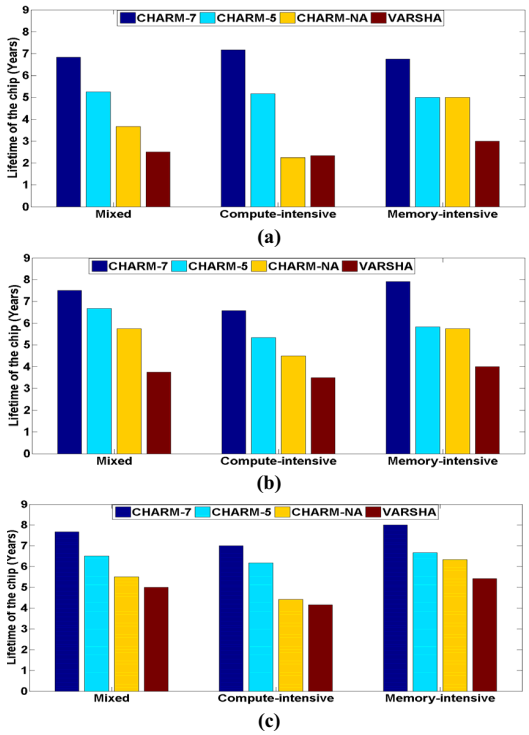

Figure 5: Comparison of lifetimes of the chip for different frameworks across different workloads and arrival rates: (a) Arrival-rate-1, inter-app duration is 1.4s (b) Arrival-rate-2, inter-app duration is 2.8s (c) Arrival-rate-3, inter-app duration is 5s

Figure 6 shows the total number of applications executed over the lifetime of the chip for the four frameworks. At all the three arrival rates, *CHARM-7* executes a higher number of applications than any other framework, achieving up to 2× improvement compared to *CHARM-NA* and up to 6× improvement compared to *VARSHA,* in the number of applications executed. This is due primarily to the higher lifetime constraint for *CHARM-7* and the ability of our proposed heuristics to manage circuit aging to meet this constraint, while reducing the number of dropped applications compared to *CHARM-NA*

and *VARSHA*. *CHARM-5* executes 2× more applications than *CHARM-NA* and *VARSHA* for compute-intensive and mixed workloads but gives results comparable to *CHARM-NA* for memory-intensive workloads when the arrival rate is high (figure 6(a)). This is because, although memory-intensive apps consume less power, they run for longer durations and have shorter *app-slack* times compared to compute-intensive apps. When the arrival rate is high, *CHARM-5* drops higher number of applications in the latter part of its lifetime owing to its inability to find a mapping region that meets application deadlines while preserving the lifetime at the same time. This leads to a similar number of applications executed in *CHARM-5* and *CHARM-NA*. However, when the arrival rate is low (figure 6(c)), *CHARM-5* surpasses the applications executed by *CHARM-NA* and *VARSHA* by over 2.5×. *CHARM-7* executes 10-40% higher number of applications compared to *CHARM-5,* due primarily to the longer lifetime achieved with *CHARM-7* which provides more opportunities to execute greater number of applications. *As CHARM-NA* prioritizes performance by executing applications at higher V_{dd} and DoP, and *VARSHA* executes applications at very high V_{dd} to safeguard the applications from soft-errors in the absence of checkpointing and rollback recovery, both frameworks suffer from relatively lower lifetimes and application execution counts. Running at higher V_{dd} also leads to higher power dissipation and fewer number of applications running on the CMP to avoid violating DS-PB constraints. As a result, in *CHARM-NA* and *VARSHA*, the waiting time in the service queue is much higher, and a larger number of applications get dropped due to missed deadlines.

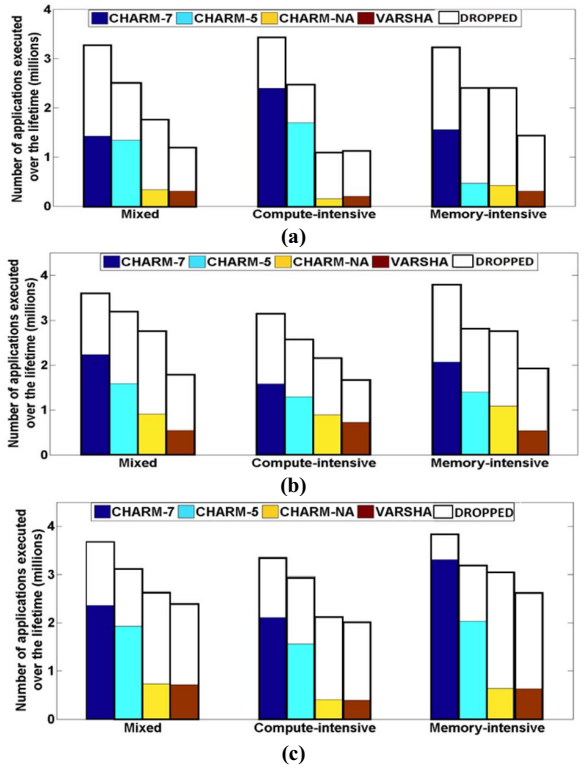

Figure 6: Results comparing the number of applications executed by different frameworks across different workloads and arrival rates. (a) Arrival-rate-1, inter-app duration is 1.4s (b) Arrival-rate-2, inter-app duration is 2.8s (c) Arrival-rate-3, inter-app duration is 5s

Figure 7 shows the average power dissipated per application by the four different frameworks. When the arrival rate is high (figure 7(a)), *CHARM-7* dissipates 50-80% less power per application than both *CHARM-NA* and *VARSHA* with different workload types. *CHARM-5* achieves 50-80% improvement in power compared to both CHARM-NA and VARSHA. This is because of the queue-pressure threshold (Section V.B), which does not allow high DoPs for applications, leading to more number of apps being mapped simultaneously and lesser average power dissipated per application, for *CHARM-7* and *CHARM-5*. When the arrival-rate is moderate (figure 7(b)), *CHARM-7* achieves 70% improvement compared to *CHARM-NA* and 80% improvement compared to *VARSHA,* and *CHARM-5* achieves an improvement of 25% compared to *CHARM-NA* and 30% compared to *VARSHA,* for compute intensive workloads. However, both *CHARM-7* and *CHARM-5* achieve only 2-5% improvement for memory intensive workloads compared to *CHARM-NA* and *VARSHA*. This is because both *CHARM-7* and *CHARM-5* have sufficient power budget and *app-slack* time for these workloads to map many of the applications with higher V_{dd} and DoP, similar to *CHARM-NA* and *VARSHA*. When the arrival rate is low (figure 7(c)), *CHARM-7* dissipates around over 2× more power per application than *CHARM-5* and up to 15% more power than *CHARM-NA*. In order to not significantly exceed the target lifetime constraint of the chip, *CHARM-7* maps applications more aggressively towards the latter part of its lifetime, with higher DoP and V_{dd}. This raises the average power dissipated per application mapped. *CHARM-NA* dissipates higher power than both *CHARM-5* and *CHARM-7*, because of its aggressive mapping of applications with very high V_{dd} and DoP. *CHARM-NA* and *VARSHA* dissipate power in a similar manner, within 3-8% of their respective powers except at the higher arrival rates of compute intensive workloads where CHARM-NA dissipates the highest power per application.

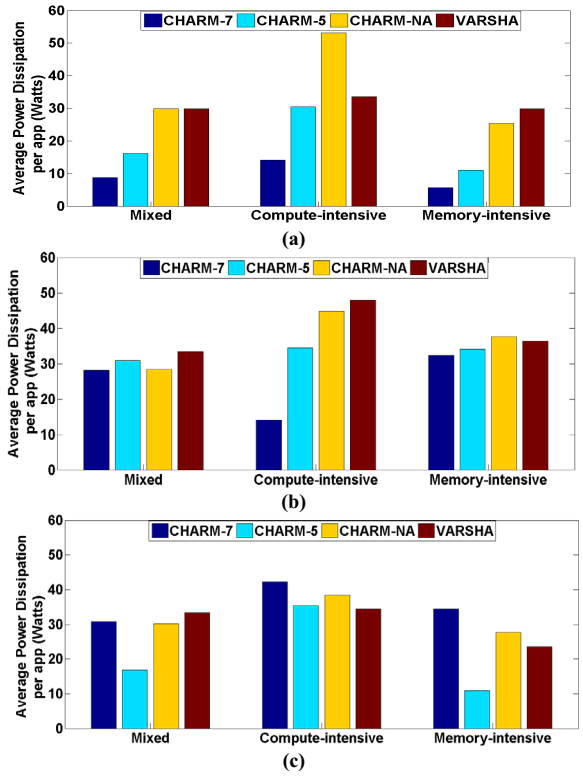

Figure 7: Average power consumed by applications executed on different frameworks across different workloads and arrival rates. (a) Arrival-rate-1, inter-app duration is 1.4s (b) Arrival-rate-2, inter-app duration is 2.8s (c) Arrival-rate-3, inter-app duration is 5s

Lastly, we analyzed the state of the chip at the end of the lifetime with the four frameworks to understand the implications of their chosen runtime resource management strategies. Table 1 shows the total number of functional cores remaining at the end of the lifetime for different workload types. The results shown are for a single arrival rate (Arrival-rate-1) for brevity. *VARSHA* reaches the end of the lifetime much before all of its cores are degraded. This is because *VARSHA*

maps applications at higher V_{dd} and frequency to satisfy reliability constraints. However, the resulting chip-aging-profile with *VARSHA* creates fragmentation, making it impossible to obtain contiguous mapping regions on the chip that meet application deadline constraints, for low or high DoP values. All the *CHARM* frameworks, however, utilize the chip till most of the cores are degraded, thereby utilizing the chip very effectively. On average, *CHARM-7* has 5×, *CHARM-5* has 2.5× and *CHARM-NA* has 2× better chip utilization vs. *VARSHA* by the time of the end of the chip lifetime.

TABLE 1: Number of functional cores at the end of the lifetime

	CHARM-7	CHARM-5	CHARM-NA	VARSHA
Mixed	10	15	17	42
Compute	9	14	20	55
Memory	11	17	17	54

VII. CONCLUSIONS

In this paper we proposed a novel runtime framework called *CHARM* that aims to maximize the number of applications executed reliably while meeting their performance deadlines without violating the dark-silicon power constraints over a given chip target lifetime. Our experiments show that CHARM enables up to 2.5× improvement in the lifetime, up to 6× improvement in number of applications executed and 5× improvement in efficiently using the cores during the lifetime of the chip compared to the state-of-the-art on reliability aware runtime application mapping.

ACKNOWLEDGEMENTS

This research is supported by grants from SRC, NSF (CCF-1252500, CCF-1302693), and AFOSR (FA9550-13-1-0110).

REFERENCES

[1] A. Kaouache, et al., "Analytical method to evaluate soft error rate due to alpha contamination," IEEE Trans, on Nuclear Science, 60(6), Dec. 2013.
[2] S. Abe, et al., "Neutron-induced soft error analysis in MOSFETs from a 65 to a 25 nm design rule using multi-scale monte-carlo simulation method," Proc. IEEE IRPS, pp.SE.3.1-SE.3.6, Apr. 2012.
[3] V. B. Kleeberger et al., "A compact model for NBTI degradation and recovery under use-profile variations and its application to aging analysis of digital integrated circuits," Microelectronics Reliability, 54(6), 2014.
[4] D. Bergstrom et al., "Intel's 45 nm CMOS technology," Intel Technology Journal, vol. 12(2), June 2008.
[5] J. Lee, et al., "Optimizing total power of many-core processors considering voltage scaling limit and process variations," Proc. ACM/IEEE ISLPED, pp: 201-206, July 2009.
[6] S. Borkar "Design perspectives on 22nm CMOS and beyond," IEEE/ACM Design Automation Conference (DAC), pp: 93-94, July 2009.
[7] J. Allred, et al., "Designing for dark silicon: a methodical perspective on energy efficient systems," ACM/IEEE ISLPED, pp: 255-260, July 2012.
[8] B. Raghunathan, et al., "Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors," Proc. ACM/IEEE DATE, pp: 39-44, Mar. 2013.
[9] N. Kapadia, et al., "VARSHA: Variation and reliability-aware application scheduling with adaptive parallelism in the dark-silicon era," ACM/IEEE DATE, pp: 1060-1065, Mar. 2015.
[10] S. Hari et al., "mSWAT: Low-Cost Hardware Fault Detection and Diagnosis for Multicore Systems," Micro 2009.
[11] A. Das, et al., "Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs," ACM/IEEE DATE, pp: 1-6, Mar. 2014.
[12] M. Haque, et al., "Energy-aware task replication to manage realiability for periodic real-time applications on multicore platforms," Proc. IGCC, 2013
[13] Y. Guo, et al. "Generalized standby-sparing techniques for energy-efficient fault tolerance in multiprocessor real-time systems." Proc. IEEE RTCSA, 2013.
[14] A. Tiwari, et al., "Facelift: Hiding and slowing down aging in multicores," MICRO Nov. 2008.
[15] N. Kapadia, V. Y. Raparti, S. Pasricha, "ARTEMIS: An Aging-Aware Run-Time Application Mapping Framework for 3D NoC based Chip Multiprocessors," IEEE/ACM Networks-on-Chip (NOCS), 2015.
[16] S. Sarangi et al., "VARIUS: A model of process variation and resulting timing errors for microarchitects," IEEE TSM, (21)1, 2008.
[17] D. Zhu, et al, "The effects of energy management on reliability in real-time embedded systems," Proc. ICCAD, Nov. 2004.
[18] J. Velamala et al., "Statistical aging under dynamic voltage scaling: a logarithmic model approach," IEEE CICC, Sept. 2012.
[19] J. Velamala et al., "Physics matters: statistical aging prediction under trapping/detrapping," DAC June 2012.
[20] Q. Han, et al. "Energy minimization for fault tolerant scheduling of periodic fixed-priority applications on multiprocessor platforms." Proc. DATE, 2015.
[21] M. Fattah et al., "Smart hill climbing for agile dynamic mapping in many-core systems," Proc. DAC, June 2013
[22] M. Arjomand et al., "Voltage-frequency planning for thermal-aware, low-power design of regular 3-D NoCs," IEEE VLSID, Jan. 2010.
[23] 3D-ICE open-source tool: http://esl.epfl.ch/3d-ice.html
[24] H. Chiu, et al. "The odd-even turn model for adaptive routing." Parallel and Distributed Systems, IEEE Transactions on 11.7 (2000): 729-738
[25] A. Bravaix, et al. "Hot-carrier acceleration factors for low power management in DC-AC stressed 40nm NMOS node at high temperature." Reliability Physics Symposium, IEEE, 2009.
[26] D. Ancajas, et al. "HCI-tolerant NoC router microarchitecture." Proc. IEEE/ACM DAC, 2013.
[27] S.V. Woo et al., "The SPLASH-2 programs: characterization and methodological characterization," ISCA, pp. 24-36, May 1995.
[28] C. Bienia et al., "The PARSEC benchmark suite: characterization and architectural implications," PACT, Oct. 2008.
[29] T.E. Carlson et al., "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," HPCA, Nov. 2011.
[30] L. Sheng et al. "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," MICRO, 2009
[31] P. Singh et al., "Dynamic NBTI management using a 45nm multi-degradation sensor," CICC Sept. 2010.