A Cross-Layer Runtime Framework for Checkpoint-based Soft-Error and Aging Management in SoCs

Conference Paper · September 2016

2 authors, including:

Venkata Yaswanth Raparti
Colorado State University

Some of the authors of this publication are also working on these related projects:

Reliability aware designs for multicore systems
A Cross-Layer Runtime Framework for Checkpoint-based Soft-Error and Aging Management in SoCs

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

Abstract—Transient faults due to single and multiple bit-flips and permanent aging effects due to Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI) gradually reduce chip reliability over time. Unfortunately, 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. In this paper, we propose a novel runtime framework (CHARM) to manage the useful chip lifetime, while also addressing transient faults and meeting dark-silicon and application performance constraints. Experimental results on a 60-core chip multiprocessor show that CHARM achieves an improvement of up to 2.5× in lifetime, up to 5× in resiliency to soft-errors, and up to 6× in number of applications executed over the chip lifetime compared to a state-of-the-art solution.

1. 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]. This phenomenon has caused newer process technologies to be significantly decreased [1]. Along with decreasing charge flip in a memory, the critical charge, which is the minimum charge needed to achieve a threshold voltage (Vt), results in higher circuit delay. Such Vt-degradation causes a slowdown of critical paths in cores and network-on-chip (NoC) routers, limiting overall system performance.

At the same time, the slowdown of power scaling with technology scaling, due to leakage and reliability concerns [3], 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) [4], chip multiprocessor (CMP) designs are becoming increasingly power-limited rather than area-limited. Runtime power-saving techniques such as dynamic voltage frequency scaling (DVFS) 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, utilizing DVFS at the core level granularity and intelligently mapping application tasks to CMP cores can yield significant benefits in terms of power, performance, and rate-of-aging trade-offs.

Additionally, recent works such as [5] 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.

However, there are intricate inter-dependencies 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). As an example, in modern power-constrained designs, CMPs are operated at lower voltage levels to save power. 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 DVFS.

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 DVFS 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 support for checkpointing and rollback in our framework. Our novel contributions in this work are summarized as follows:

- we propose a novel runtime framework (CHARM) for application mapping and DVFS 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 Vdd and execution frequency, based on queue pressure and app-sack 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 Vt degradation over the lifetime of the chip;

II. PROBLEM FORMULATION

A. Models for Reliability Estimation

Circuit-aging due to BTI leads to a degradation of the threshold voltage (Vt) under a sequence of Vdd’s used in the DVFS operation. Our analysis of circuit-aging over the CMP-lifetime is based on the long-term aging prediction model proposed in [8], which accounts for different Vdd-levels over time. HCI also leads to degradation of the threshold voltage (Vt) which can be modeled as a function of stress-time [11]. We model the maximum frequency of a core based on its supply voltage as given in [6].

We model soft error-rates (SER) as discussed in [7]. We define λ(f), as the SER at a given frequency f by the equation below:

\[ \lambda(f) = \lambda_0 \cdot 10^{\frac{d}{dVd} \log_{10}(f)} \]  

where λ0 is the SER corresponding to the highest frequency value (fmax). For compute cores we consider λ0 = 10^6 errors/sec and assume d=3 [5]. We compute the probability of one or more faults occurring over an execution period \( t \) using Eq. (2):

\[ P(f, t) = 1 - e^{-\lambda(f)t} \]  

\[ E(f, t) = \int_0^t \lambda(f) \cdot P(f, t) \cdot dt \]  

Eq. (3) gives the expected number of faults \( E(f, t) \) observed in a given time interval \([0, t]\) during which \( f \) remains constant. We compute the number of faults in any given interval \([t_1, t_2]\) as follows:

\[ E(t_1, t_2) = E(t_2, t_2) - E(t_1, t_1) \]

We model a checkpoint and rollback based error recovery mechanism as proposed in [9]. The number of checkpoints employed for a task is a function of its worst case execution time \( L \) and its deadline \( D \). CHARM decides the number of checkpoints based on the deadline
$D_i$ of the task graph to be mapped. Eq. (5) gives the optimum number
of checkpoints $n_i$ assigned to a task $i$:

$$n_i \leq 2 \frac{c_i}{D_i-T_i-r_i-s_i} - 1 \quad (5)$$

where, $D_i$ is the deadline of the task $i$, $T_i$ is the fault free task 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 $T_i/n_i$.

B. Inputs, assumptions and problem objective

We assume the following inputs to our problem:

- A 2D mesh NoC-based CMP of dimension $[dim, dim, dim]$ and
  number of tiles $N=dim \times dim \times dim$; each tile has a core and a router;
- A chip-wide dark-silicon power budget (DS-PB);
- An application task graph for each application to be executed on the
  CMP, vertices with task execution-times on compute cores and
  edges with inter-task communication-volumes;
- A set of supply voltages $V_{\text{dd}} = \{V_1, V_2, ..., V_n\}$ for each core;
- Application task graphs for the set $P = \{P_1, P_2, ..., P_m\}$ of DoPs for
  all applications; an application $i$ possesses $|P_i|$ viable DoPs;
- A set of permissible rectangular shapes of regions that an application
  can be mapped on to $\{B_1, ..., B_n\}$.

We make the following assumptions in our work:

- Applications are mapped contiguously on to non-overlapping
  rectangular shaped regions of the 2D CMP for inter-application
  isolation and optimized communication-profiles;
- Per-core granularity of DVFS is considered, to meet DS-PB and
  application performance demands, and facilitate runtime selection of
  DoP for the applications to avoid execution deadline violations;
- We assume the presence of an on-chip error detection mechanism to
  detect soft-error events and on-chip aging sensors to monitor the
  runtime $V_T$ values of individual cores and routers at the end of each
  epoch and send the values to our framework; 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;
- The CMP is rendered unusable when the chip has degraded beyond
  a set limit and an application in the service queue cannot be mapped;

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), DoP values,
and checkpoint periods, as well as a per-core DVFS 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 1: Overview of CHARM runtime app-DoP selection, reliability
aware mapping, and DVFS scheduling framework.

III. CHARM FRAMEWORK: OVERVIEW

The key aspects of our proposed framework are illustrated in figure
1. CHARM makes decisions based on the runtime input it receives from
on-chip aging sensors and app-slate time $= \{ \text{worst case execution time}
– \text{deadline} \}$ available for an application waiting in the service queue.
CHARM intelligently prioritizes between lifetime and performance
based on the available app-slate time and the observed chip
degradation profile. The framework dynamically selects the DoP,
checkpoint period, and per-core $V_{\text{dd}}$ for each application’s execution,

Figure 2: CHARM framework design-flow: (a) circuit-aging, lifetime and
epoch management (outer-loop, Section-III.A); (b) reliability aware
mapping, DVFS and app-DoP scheduling (inner-loop, Section-III.B);
blocks shown with dotted outlines are simulated models used in our work,
and are a proxy for on-chip sensors that will provide the information at
runtime in a real CMP system.

A. Circuit-aging, lifetime, and epoch management

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-slate 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 of our framework performs reliability-aware
mapping, app-DoP selection, and $V_{\text{dd}}$ selection during an epoch and is
discussed in section III.B (shown as the pink box in figure 2(a), with
an expanded view in figure 2(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 ($\Delta V_T$)’s of all cores and NoC routers on the CMP (i.e., extent
of BTI and HCI-induced circuit aging) is calculated during all of their
AT’s over the entire epoch. The computed $\Delta V_T$’s are saved and passed
on to the inner loop for reliability aware mapping, DoP and $V_{\text{dd}}$
selection (section III.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 $\Delta V_T$’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
based on runtime inputs and available slack. CHARM operates as two
nested loops, (i) circuit-aging, lifetime and epoch management (figure
2(a), outer loop); and (ii) reliability-aware application mapping, DVFS,
and application-DoP selection (figure 2(b), inner loop). These two
components are discussed in detail in the following subsections.
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) exceeds a specified threshold.

**B. Reliability aware mapping, DVFS and app-DoP selection**

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. Before mapping, \( \text{CHARM} \) assigns checkpointing periods for each application as given by Eq. (5). If the worst-case execution time cannot meet the application-slabk time using the available on-chip resources, that application is dropped from the service queue. For any application under consideration, this stage consists of two phases, (i) region-selection, app-DoP and \( V_a \) selection and (ii) communication-aware task-to-tile mapping. 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 \( \{B_1,...,B_n\} \) for that application. Our heuristic in this step utilizes the runtime \( V_T \)-degradation profile of the CMP, which is given as:

\[
\Omega = \sum_{i=1}^{k} V_T k
\]

where \( V_T \) 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 \( \Omega \) when the value is greater than a threshold \( \zeta \) 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 \( \psi \) to select the rectangular region on the CMP:

\[
\psi = \frac{V_T + \Delta V_T}{\max V_T - \min V_T} \quad \text{for all} \quad i \in \text{tile shedules}
\]

where \( V_T \) is the same as in Eq. (6) for cores within the region; and \( \min V_T \) and \( \max V_T \) are the nominal (lowest) effective \( V_T \)-value of a core with no aging. We define \( \Delta V_T \) as the maximum \( V_T \) value that the core can support (at highest \( V_a \)). In order to preserve lifetime, \( \text{CHARM} \) selects a region with the least \( \psi \) that satisfies the application’s deadline and the chip DS-PB constraints. This in effect results in the selection of the most aged-cores that still satisfy the application deadline, which helps increase the overall lifetime of the chip.

**Algorithm 1: Reliability-aware region, DoP and \( V_a \)-selection heuristic**

**Inputs:** \( V_T \)-profile, \( \{P_1,...,P_k\}, \{B_1,...,B_n\}, \{V_a,...,V_f\}, \text{DS-PB} \)

1: for each DoP in \( \{P_1,...,P_k\} \) & each \( V_a \) in \( \{V_a,...,V_f\} \) & each tile in CMP, do
2: for each shape in \( \{B_1,...,B_n\} \) do
3: list_time.insert(\( P,B,V_a \))
4: list_age.insert(\( P,B,V_a \))
5: \text{for} each shape
6: \text{// end for}
7: \text{if} (\( \Omega < \zeta \) & (app-slabk time is less than \( \tau \))
8: \( \text{ptr} = \text{list_time.begin()}\)
9: \text{// end if}
10: \text{else if} ((app-slabk is greater than \( \tau \)) or \( \Omega \geq \zeta \))
11: \( \text{ptr} = \text{list_age.begin()}\)
12: \text{// end if}
13: while(\( \text{ptr} = \text{list_time.end()}\) do{
14: \text{check if CMP meets DS-PB with the ptr} \rightarrow (\( P,B,V_a \))
15: \text{if (DS-PB constraint not met) then ptr++}
16: \text{// end while}
17: if(\( (P,B,V_a) \) is not found) drop the application
18: \text{output: a valid region to map the application, app-DoP value and } V_a \text{-level for cores in the selected region; or application being dropped}

Algorithm 1 shows the pseudo code for our region, DoP and \( V_a \) selection heuristic. The heuristic performs a search over the available per-core supply voltages \( \{V_1,...,V_n\} \), application DoPs \( \{P_1,...,P_k\} \) and the permissible mapping regions \( \{B_1,...,B_n\} \) for the application. Aging profiles \( \{V_1,...,V_n\} \) 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 finds a suitable \{DoP, mapping region and \( V_a \)\} combination for an application under consideration for mapping to the CMP.

The mapping heuristic does an exhaustive search over combinations of all the DoP \( \{P_i\} \), \( V_a \) 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 \( (P_i,B_i,V_a) \) combination that meets the application deadline. The second list, list_age, is in the increasing order of the region’s aging profile \( \psi \), given by Eq. (7). The heuristic then iterates for a suitable candidate from the list_time if there is enough app-slabk time and the CMP has degraded less than threshold \( \zeta \) (lines 7-9), while meeting the DS-PB constraints. In all other cases, the heuristic 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, performance and reliability constraints, 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 we trim down the permissible DoPs for mapping an application when the queue pressure is above a threshold \( \chi' \).

We now present the theoretical time complexity of our heuristic. Our region/\( V_a \)/DoP selection heuristic effectively runs in linear-time complexity with respect to the number of tiles, \( N \): \( O(n \times |D| \times |S| \times N) \), where \( |D| \) is the permissible DoPs, \( |S| \) is the permissible \( V_a \) levels for the cores, \( n \) is the number of admissible shapes, and \( c \) is the DoP of the application (all of these are relatively small integers). Thus our heuristic is 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 [10]) suitable for use at runtime, which aims to minimize communication between cores.

**IV. EXPERIMENTAL STUDIES**

**A. Simulation setup**

We conducted experiments on 13 different parallel applications from the SPLASH-2 [12] and PARSEC [13] benchmark suites. 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. We categorized the 13 benchmarks into two groups: (i) memory-intensive benchmarks - (cholesky, fft, radix, raytrace, dedup, canneal, vips); and (ii) compute-intensive benchmarks – (swaptions, fluidanimate, streamcluster, blackscholes, radix, bodtrack, radiosity). We employ three types of application sequence groups, (memory-intensive, compute-intensive and mixed), each having 100 randomly ordered application-instances selected from the respective group.

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 10x6 mesh layout. The \( V_a \) values supported by each tile (core + router) are between 0.75V-1V, in steps of 0.05V. The dark-silicon power budget (DS-PB) is assumed to be 80W. 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.

**B. Simulation Results**

We compare our CHARM framework against an enhanced version of a prior work VARSHA [5] that tries to optimize the energy and performance of a CMP while meeting dark-silicon power constraints as well as satisfy reliability and performance constraints. We explore
three variants of our CHARMM framework: CHARMM-5, which is designed for a target lifetime of 5 years; CHARMM-7, which is designed for a target lifetime of 7 years; and CHARMM-NA, which has no target lifetime. CHARMM-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, and the average power dissipated by applications, for the four frameworks. Figure 3 shows the lifetimes of the CMP for the different frameworks. CHARMM-7 and CHARMM-5 are designed to achieve their target lifetimes of 7 and 5 years respectively. This is made possible by changing the threshold value $\zeta$ for different target lifetimes. For CHARMM-7, $\zeta$ is empirically derived to be approximately 21.5V for inter-app-duration of 1.4s. Similarly for CHARMM-5 it is approximately 27V. Both CHARMM-7 and CHARMM-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, CHARMM-NA optimizes primarily for performance while VARSHA optimizes for energy, leading to their lowered lifetimes. In particular, CHARMM-7 achieves 50-100% improvement in lifetime compared to CHARMM-NA, and up to 2x improvement compared to VARSHA.

![Figure 3: Comparison of lifetimes of the chip for different frameworks across different workloads with inter-app-duration of 1.4s](image)

![Figure 4: Number of applications executed by different frameworks across different workloads with inter-app-duration of 1.4s](image)

![Figure 5: Average power for applications executed on different frameworks across different workloads with inter-app-duration of 1.4s](image)

Figure 4 shows the total number of applications executed over the lifetime of the chip for the four frameworks. CHARMM-7 achieves up to 2x improvement compared to CHARMM-NA and up to 6x improvement compared to VARSHA, in the number of applications executed. This is due primarily to the higher lifetime constraint for CHARMM-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 CHARMM-NA and VARSHA. CHARMM-5 executes 2x more applications than CHARMM-NA and VARSHA for compute-intensive and mixed workloads but gives results comparable to CHARMM-NA for memory-intensive workloads. 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. As CHARMM-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. This also leads to higher power dissipation and violating DS-PB constraints. As a result, in CHARMM-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.

Lastly, figure 5 shows the average power dissipated per application by the four different frameworks. CHARMM-7 and CHARMM-5 dissipates 50-80% less power per application than both CHARMM-NA and VARSHA with different workload types. This is because of the higher number of applications being mapped simultaneously and lesser average power dissipated for the mapped applications, with CHARMM-7 and CHARMM-5. CHARMM-NA dissipates higher power than both CHARMM-5 and CHARMM-7 because of its aggressive mapping of applications with very high $V_{dd}$ and DoP. CHARMM-NA and VARSHA dissipate power in a similar manner, within 3-8% of their respective powers except at higher arrival rates of compute intensive workloads where CHARMM-NA dissipates the highest power per application.

V. CONCLUSIONS

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

REFERENCES


