Variation-Aware Task Allocation and Scheduling for Improving Reliability of Real-Time MPSoCs

Junlong Zhou†, Tongquan Wei‡, Mingsong Chen‡, X. Sharon Hu§, Yue Ma§, Gongxuan Zhang†, Jianming Yan‡
†School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
‡School of Computer Science and Software Engineering, East China Normal University, Shanghai 200241, China
§Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46656, USA

***Abstract*—Both soft-error reliability (SER) due to transient faults and lifetime reliability (LTR) due to permanent faults are key concerns in real-time MPSoCs. Existing works have investigated related problems, however, most of them only focus on one of the two reliability concerns. A few efforts do consider both types of reliability together, but ignore the impacts of hardware- and application-level variations on reliability, thus are not applicable to state-of-the-art MPSoCs under variations.**

In this paper, we focus on increasing SER without sacrificing LTR since transient faults occur much more frequently than permanent faults. Specifically, we propose a novel task allocation and scheduling scheme to maximize SER while satisfying a LTR constraint for soft real-time MPSoCs. Considering that SER is the objective while LTR is a constraint in our problem, and LTR is highly related to core temperature profiles, we dedicate to investigating the effects of variations in core soft-error rate, task vulnerability to soft errors, and task execution time on SER. To the best of our knowledge, our work is the first attempt that jointly handles the two reliability issues as well as taking into account the effects of variations on reliability. Experimental results show that our scheme improves the SER by up to 66% as compared to a number of representative existing approaches while meeting the same LTR constraint.

***Index Terms*—Soft-error reliability; Lifetime reliability; Variations; Real-time MPSoC systems.**

I. Introduction

The advance of technology scaling enables the integration of multiple processing elements, memory hierarchies, dedicated hardware, and I/O components on a single silicon die to form a multiprocessor system-on-chip (MPSoC). Due to the advantages of powerful parallel processing capability, higher computing density, and lower clock frequencies, MPSoCs have replaced uniprocessors to become the mainstream for microprocessors in various application domains. The distinct features of MPSoCs can be exploited to meet the stringent design requirements of emerging real-time applications.

Many real-time embedded systems running on MPSoCs are deployed in critical applications and harsh environments, thus are expensive as well as inconvenient to repair and replace. To maintain quality of service and reduce cost of repairing/replacing an entire system, soft-error reliability (SER) due to transient faults and life-time reliability (LTR) due to permanent faults are imperative design concerns. In this paper, we are interested in solving the reliability concerns for real-time MPSoCs. As transient faults occur much more frequently than permanent faults [1], we focus on the problem of maximizing SER while satisfying the constraints of real-time requirements, an upper bound on LTR, and peak temperature limit. Such problems can be found in real-world applications such as mobile devices and in-vehicle infotainment systems [2].

This work was partially supported by Natural Science Foundation of China (Grant Nos. 61272420 and 61672230), Shanghai Municipal Natural Science Foundation (Grant No. 16ZR1409000), and U.S. NSF (Grant No. CNS-1319904). J. Zhou is the corresponding author, email: jlzhou@njust.edu.cn.

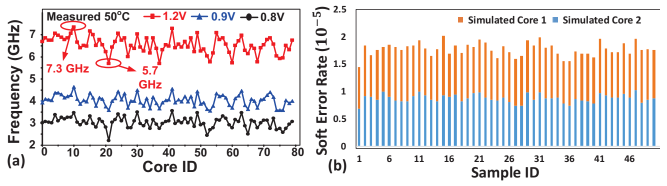

Fig. 1: Core-to-core variations in (a) maximum frequency due to manufacturing process variation [11], and (b) soft-error rate.

Considerable research efforts have been devoted to the design of reliability-aware real-time systems. However, most of them either focus on SER [1], [3], [4] or LTR [5]–[7]. A few recent works [8]–[10] have examined both SER and LTR together. To maximize system availability, Zhou et al. [8] designed a static framework that balances SER and LTR by determining the replication and frequency of tasks. Ma et al. [9] developed a dynamic heuristic for enhancing SER and LTR of real-time systems running on "Big-Little" type MPSoCs. Kim et al. [10] introduced two energy and lifetime optimization techniques that adopt DVFS-aware reliability model and Q-learning-based method for many-core microprocessors considering both SER and LTR. However, the effects of hardware- and application-level variations on reliability are not taken into account in these works.

In this paper, we study the effects of variations in core soft-error rate, task vulnerability to soft errors, and task execution time on SER. Based on the study, we design a scheme that maximizes SER under a constraint of LTR. We leave the investigation of the effects of variations at hardware and application levels on LTR to future work since LTR is set as a constraint in our problem and is decided by core temperatures.

II. Impacts of Core and Task Variations on SER

Fig. 1(a) shows that core-to-core frequency variations in an Intel's 80-(homogeneous) core test chip are 28% at 1.2V and 59% at 0.8V [11]. Since soft-error rates of cores are highly dependent on core frequencies, core-to-core frequency variations must lead to core-to-core soft-error rate variations. In addition to homogeneous cores, we investigate the soft-error rate variations of heterogeneous cores using Monte Carlo sim-

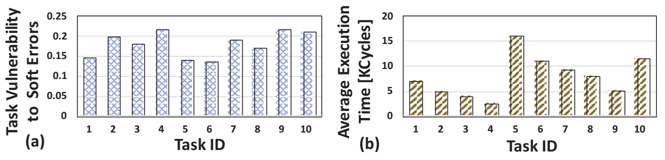

Fig. 2: Task-to-task variations in (a) vulnerability to soft errors, and (b) execution time in terms of number of clock cycles [3].

ulation[1]. The results are illustrated in Fig. 1(b), in which each data is averaged over 200 Monte Carlo samples. Clearly, the soft-error rates of two heterogeneous cores are quite different. Besides the hardware-level variations, different applications also exhibit distinct variations in vulnerability to soft errors and execution time. Fig. 2(a) and Fig. 2(b) [3] show the task-to-task variations in vulnerability to soft errors, and average execution time in terms of number of clock cycles, where ten tasks are real-world benchmarks DCT, FIR, SAD, SATD, BubleSort, MergeSort1, MergeSort2, QuickSort1, QuickSort2, SelectionSort [3]. Such task-to-task variations are due to the diverse data and control flow properties.

Since the system-level SER of running tasks on cores is a function of hardware-level soft-error rate, task vulnerability, and task execution time, the variations shown in Fig. (1) and Fig. (2) must have non-negligible impacts on SER and need to be considered in the design for SER optimization.

III. Our Contributions

In this paper, we aim to solve the problem of maximizing SER for real-time MPSoC systems under the constraints of LTR, peak temperature, and task deadline. We make the following major contributions.

1) We investigate the impacts of variations at hardware and application levels in terms of core soft-error rate, task vulnerability and execution time on SER.
2) We develop a variation-aware task allocation approach that exploits the diverse reliability properties of cores and tasks to maximize SER.
3) We present a cross entropy-based task scheduling approach that utilizes the power of cross entropy for optimization to minimize SER degradation incurred by meeting system design constraints.

IV. System Model

In this section, we describe the architecture and application model as well as SER and LTR models.

[1]We take 10,000 Monte Carlo samples to compare the soft-error rate of two simulated heterogeneous cores. The heterogeneity of simulated cores is realized by assuming that cores consist of different components with various proportions. Specifically, simulated core 1 is made up of 6T SRAM cell, Latch, and NAND2 while simulated core 2 is made up of 8T SRAM cell, Latch, and NAND2 [12]. For each core, the respective proportions of 3 components are represented by ψ_1, ψ_2, ψ_3 and hold for $\psi_1+\psi_2+\psi_3=1$, and the respective architecture vulnerability factors of 3 components are represented by AVF_1, AVF_2, AVF_3 and take the value of $(0,1]$. Using the sum-of-error rate model [13] and the error rate λ_1, λ_2, λ_3 of 3 components measured in [12], the soft-error rate of a core can be estimated as $\psi_1 \cdot AVF_1 \cdot \lambda_1 + \psi_2 \cdot AVF_2 \cdot \lambda_2 + \psi_3 \cdot AVF_3 \cdot \lambda_3$. Without loss of generality, Monte Carlo simulation is used to produce samples by randomly setting the value of ψ_1, ψ_2, ψ_3 and AVF_1, AVF_2, AVF_3. Two produced cores with different ψ_1, ψ_2, ψ_3, AVF_1 and same AVF_2, AVF_3 constitute one sample of Monte Carlo simulation.

A. Architecture and Application

Consider an MPSoC system $\mathcal{C}$ consisting of m cores $\{\mathcal{C}_1, \cdots, \mathcal{C}_m\}$ and a task set Γ consisting of n independent periodic tasks $\{\tau_1, \cdots, \tau_n\}$ with soft real-time constraints. Each task τ_i $(1 \leq i \leq n)$ is characterized by a quadruplet $\{p_i, d_i, wc_i, TVF_i\}$, where p_i is the period, d_i is the relative deadline, wc_i is the worst-case execution time in cycles, and TVF_i is the task vulnerability factor indicating the probability that a transient fault at the hardware level ultimately leads to a program failure at the task level [3]. In general, the period of a task is equal to its deadline, i.e., $p_i = d_i$. The hyper-period of the task set, denoted by HP, is the lowest common multiple of all task periods $\{p_1, \cdots, p_n\}$.

The task set is periodically executing on the MPSoC, each core of which is dynamic frequency scaling enabled and supports a discrete set of frequencies varying from the minimum frequency to the maximum frequency. Denoting the minimum and maximum frequency of core $\mathcal{C}_j$ $(1 \leq j \leq m)$ as $F_{j,min}$ and $F_{j,max}$, respectively, the frequency $F_{j,k}$ $(1 \leq k \leq \gamma_j)$ of core $\mathcal{C}_j$ satisfies $F_{j,min} = F_{j,1} < \cdots < F_{j,k} < \cdots < F_{j,\gamma_j} = F_{j,max}$, where γ_j is the number of frequency levels supported by core $\mathcal{C}_j$. For the sake of easy presentation, let $f_j \in \{F_{j,1}, \cdots, F_{j,\gamma_j}\}$ denote the operating frequency of core $\mathcal{C}_j$. With respect to SER, cores are diverse in their fault rates and tasks are diverse in their vulnerability and durations.

B. Soft-Error Reliability (SER)

Transient faults can be modeled using a Poisson distribution with the average fault arrival rate, which represents the expected number of failures occurring per second and increases as the frequency decreases [3]. Let $\lambda(f_j)$ denote the raw fault rate of core $\mathcal{C}_j$ at frequency f_j, it then can be expressed as

$$\lambda(f_j) = \lambda_{F_{j,max}} \cdot 10^{\frac{F_{j,max}-f_j}{\Delta}}, \tag{1}$$

where $\lambda_{F_{j,max}}$ is the fault rate at the maximum frequency $F_{j,max}$, and Δ is a parameter that shows how the fault rate increases with frequency decrease. Considering the task error probability, the ultimate fault rate of executing task τ_i on core $\mathcal{C}_j$ at frequency f_j is then calculated as $\lambda(f_j) \cdot TVF_i$.

The SER of a task is defined as the probability of its successful execution without the occurrence of transient faults, and is decided by the exponential failure law. Using the exponential distribution assumption, the SER of task τ_i running on core $\mathcal{C}_j$ during the hyper-period HP is expressed as

$$R(\tau_i, \mathcal{C}_j) = e^{-\lambda(f_j) \cdot TVF_i \cdot \frac{wc_i}{f_j} \cdot \frac{HP}{p_i}}, \tag{2}$$

where wc_i/f_j is the execution time of task τ_i at frequency f_j and HP/p_i is the number of task instances of τ_i during the hyper-period. The system SER at a hyper-period is calculated as the product of reliabilities of all individual tasks, i.e.,

$$R_{sys} = \prod_{\tau_i \in \Gamma} \prod_{\mathcal{C}_j \in \mathcal{C}} R(\tau_i, \mathcal{C}_j). \tag{3}$$

C. Lifetime Reliability (LTR)

The increase in power density leads to elevated operating temperature and frequent temperature variations, which accelerate chip wear-out due to electromigration (EM), time dependent dielectric breakdown (TDDB), stress migration (SM), and thermal cycling (TC). Such accelerated wear-outs eventually result in permanent faults occurring earlier and reduce system lifetime. Mean time to failure (MTTF) is typically used to quantify LTR, which depends on multiple wear-out effects [6]. We focus on the four failure mechanisms in this paper. Other failure mechanisms can be incorporated using the sum-of-fault rate model [9]. We leverage the system-level modeling tool [7] to estimate LTR when considering the four failure mechanisms. The tool integrates three levels of models, i.e., device-, component- and system-level models. At the device level, the LTR models due to the four mechanisms are established. At the component level, each failure mechanism is assumed to obey a specific distribution. Based on the device-level LTR models and assumed distributions, the component-level MTTF is calculated. Then using the component-level LTR as input, the system-level MTTF is obtained by Monte Carlo simulation.

V. Framework to Maximize System-Level SER

In this section, we first formally define the SER maximization problem and then briefly describe our two-stage solution.

A. SER Maximization Problem

We address the reliability concerns for real-time systems running on MPSoCs. To be specific, we solve the following problem. Given a set Γ of n periodic real-time tasks and a set $\mathcal{C}$ of m cores, design a reliability-aware task allocation and scheduling scheme that maximizes SER while satisfying the real-time requirements, an upper bound on LTR, and the constraint on temperature. The problem is formulated as

$$\begin{aligned} \text{max:} \quad & R_{sys} = \prod_{\tau_i \in \Gamma} \prod_{\mathcal{C}_j \in \mathcal{C}} R(\tau_i, \mathcal{C}_j) \\ \text{s.t. :} \quad & ET(\tau_i) \leq d_i, \ \forall i = 1, 2, \cdots, n \qquad (4) \\ & MTTF(\mathcal{C}_j) \geq MTTF_{th}, \ \forall j = 1, 2, \cdots, m \qquad (5) \\ & T_{peak}(\mathcal{C}_j) \leq T_{th}, \ \forall j = 1, 2, \cdots, m. \qquad (6) \end{aligned}$$

The first constraint captures the real-time requirement that each task needs to be finished before deadline, where $ET(\tau_i)$ is the execution time of τ_i. The second constraint requires the MTTF of each core, $MTTF(\mathcal{C}_j)$, should be no less than a threshold $MTTF_{th}$. The last constraint is introduced to ensure that the peak temperature of each core, $T_{peak}(\mathcal{C}_j)$, cannot exceed a temperature limit T_{th}. For soft real-time systems like infotainment systems, temporarily violating the first two constraints is acceptable, but the last constraint must be satisfied to avoid the timing faults resulting from overheating.

B. Overview of Our Two-Stage Solution

The system-level SER given in Eq. (3) can be re-written as

$$\begin{aligned} R_{sys} &= \prod_{\tau_i \in \Gamma} \prod_{\mathcal{C}_j \in \mathcal{C}} R(\tau_i, \mathcal{C}_j) = \prod_{\tau_i \in \Gamma} \prod_{\mathcal{C}_j \in \mathcal{C}} e^{-\lambda(f_j) \cdot TVF_i \cdot \frac{wc_i}{f_j} \cdot \frac{HP}{p_i}} \\ &= e^{-\sum_{\tau_i \in \Gamma} \sum_{\mathcal{C}_j \in \mathcal{C}} \lambda(f_j) \cdot TVF_i \cdot \frac{wc_i}{f_j} \cdot \frac{HP}{p_i}} \\ &= e^{-HP \cdot \sum_{j=1}^{m} \left(\frac{\lambda(f_j)}{f_j} \cdot \sum_{\tau_i \in \Gamma_j} \frac{TVF_i \cdot wc_i}{p_i} \right)}, \qquad (7) \end{aligned}$$

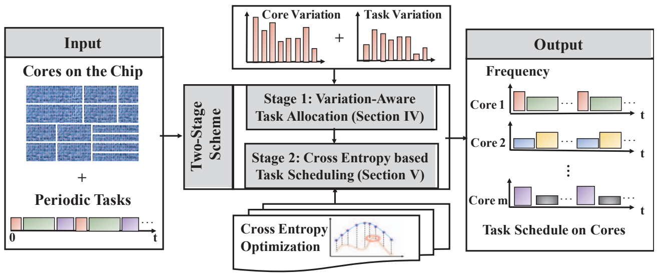

Fig. 3: Overview of our proposed two-stage scheme.

where Γ_j is the set of tasks allocated to core $\mathcal{C}_j$. Clearly, the system SER depends on the task-to-core allocation (represented by Γ_j) and the frequency setups (represented by f_j).

We solve the SER maximization problem at two stages. In the first stage, we develop a variation-aware task allocation approach to maximize system SER by exploiting core and task variations. The task allocation approach assumes that all the tasks are to be executed at the highest frequencies of their respectively allocated cores for the sake of decreasing transient fault rates thus increasing SER. However, high frequencies cause high temperatures, which may lead to violation of MTTF and temperature constraints. Thus, we check the two constraints in the second stage. If the highest frequencies of individual cores are safe to use for improving SER, the SER-optimum task allocation and the highest frequencies are output as the final task schedule[2]. Otherwise, the optimality of SER is traded for a larger MTTF and a lower peak temperature by scaling down the operating frequencies of tasks. Since our objective is to maximize SER, the less sacrifice of SER optimality is made for satisfying the constraints, the higher SER can be obtained. It is well known that our frequency scaling (or called selection) in the second stage is an NP-hard combinatorial optimization problem, which can be successfully solved by a powerful tool, cross entropy method [14]. Therefore, we design a cross entropy-based approach in the second stage to address our task frequency selection problem of minimizing SER degradation and satisfying the design constraints. The overview of our scheme is shown in Fig. 3.

VI. Variation-aware Task Allocation

This section analyzes the optimality of allocating tasks to multiple cores in terms of SER, presents a theorem on the optimum task allocation, and develops a task-to-core allocation heuristic based on the theorem.

A. Optimality Analysis of Task Allocation

Assuming that tasks are executed at the maximum frequencies of their respectively assigned cores, the system-level SER given in Eq. (7) can be formulated as

$$R_{sys} = e^{-HP \cdot \sum_{j=1}^{m} \left(\frac{\lambda(F_{j,max})}{F_{j,max}} \cdot \sum_{\tau_i \in \Gamma_j} \frac{TVF_i \cdot wc_i}{p_i} \right)}, \qquad (8)$$

where $F_{j,max}$ is the maximum frequency of core $\mathcal{C}_j$. Clearly, the system SER R_{sys} is maximal when the term $\sum_{j=1}^{m} \left(\frac{\lambda(F_{j,max})}{F_{j,max}} \cdot \sum_{\tau_i \in \Gamma_j} \frac{TVF_i \cdot wc_i}{p_i} \right)$ is minimized. Thus,

[2]The task execution order can be determined by adopting one of classical priority assignment algorithms (e.g., RM) widely used in real-time systems.

the term can be defined as a metric used to characterize the system SER. It is represented by $Metric_{SER}$ and can be formulated into the product of two vectors $\mathcal{U}$ and $\mathcal{V}$, i.e.,

$$Metric_{SER} = \sum\nolimits_{j=1}^{m} \Big(\frac{\lambda(F_{j,max})}{F_{j,max}} \cdot \sum\nolimits_{\tau_i \in \Gamma_j} \frac{TVF_i \cdot wc_i}{p_i} \Big) = \frac{\lambda(F_{1,max})}{F_{1,max}} \cdot \sum\nolimits_{\tau_i \in \Gamma_1} \frac{TVF_i \cdot wc_i}{p_i} + \cdots + \frac{\lambda(F_{j,max})}{F_{j,max}} \cdot \sum\nolimits_{\tau_i \in \Gamma_j} \frac{TVF_i \cdot wc_i}{p_i} + \cdots + \frac{\lambda(F_{m,max})}{F_{m,max}} \cdot \sum\nolimits_{\tau_i \in \Gamma_m} \frac{TVF_i \cdot wc_i}{p_i} = \mathcal{U}_1 \cdot \mathcal{V}_1 + \cdots + \mathcal{U}_j \cdot \mathcal{V}_j + \cdots + \mathcal{U}_m \cdot \mathcal{V}_m = \mathcal{U} \cdot \mathcal{V} \quad (9)$$

where $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m]$ and $\mathcal{V} = [\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_m]^T$.

Vector $\mathcal{U}$ captures core dependent parameters, where $\mathcal{U}_j = \frac{\lambda(F_{j,max})}{F_{j,max}}$ is referred to as the vulnerability index of core $\mathcal{C}_j$. Vector $\mathcal{V}$ captures task related parameters, where $\mathcal{V}_j = \sum_{\tau_i \in \Gamma_j} v_i = \sum_{\tau_i \in \Gamma_j} \frac{TVF_i \cdot wc_i}{p_i}$ is referred to as the vulnerability index of task set Γ_j and $v_i = \frac{TVF_i \cdot wc_i}{p_i}$ is referred to as the vulnerability index of task τ_i. According to these definitions, we can draw the following conclusion.

- Given a task set Γ, the sum of vulnerability index of all the tasks in the set is a constant.
- Given an MPSoC system $\mathcal{C}$, vector $\mathcal{U}$ can be readily derived based on the core parameters while vector $\mathcal{V}$ is not and depends on task-to-core allocation.
- The core having smaller vulnerability index is more reliable and the task having larger vulnerability index is more vulnerable to soft errors.

Based on the above, we introduce a theorem below to show that the system SER metric $Metric_{SER}$ is minimized (thus R_{sys} is maximized) when the core having smaller vulnerability index ends up with the set of its allocated tasks having larger vulnerability index. In other words, the system SER is maximal when the unreliable tasks are executed on the reliable cores. We omit the proof of the theorem due to page limit.

Theorem 1: *If the core vulnerability index in $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_m]$ satisfy $\mathcal{U}_1 \leq \mathcal{U}_2 \leq \cdots \leq \mathcal{U}_m$, and the sum of the corresponding task set vulnerability index in $\mathcal{V} = [\mathcal{V}_1, \mathcal{V}_2, \cdots, \mathcal{V}_m]^T$ is constant, then the system SER R_{sys} given in Eq. (7) is minimized if $\mathcal{V}_1 \geq \mathcal{V}_2 \geq \cdots \geq \mathcal{V}_m$ holds.*

B. Our Task Allocation Heuristic

In this work, we propose a task allocation heuristic which is motivated by Theorem 1, that is, allocating the unreliable tasks to the reliable cores can maximize the system SER. The heuristic operates as follows. Tasks in the set with the maximum vulnerability index are allocated to the core with the minimum vulnerability index, and tasks in the set with the next maximum vulnerability index are allocated to the core with the next minimum vulnerability index. This process repeats until all sets of tasks are allocated to individual cores.

The pseudo code of our heuristic is given in Alg. 1. Inputs to the algorithm are task set Γ and core set $\mathcal{C}$. The vulnerability index of m cores are supposed to satisfy $\mathcal{U}_1 \leq \mathcal{U}_2 \leq \cdots \leq \mathcal{U}_m$. Alg. 1 first initializes the sets of tasks allocated to cores to empty (lines 1-2), then determines these task sets by iteratively allocating tasks having larger vulnerability index to cores having smaller vulnerability index (lines 4-13). In each round of iteration, the vulnerability index of tasks in set Γ are calculated (lines 5-6) and the tasks are sorted in the decreasing order of the vulnerability index using Heapsort (line 7). A temporary task set Γ' is created and used to test the feasibility of allocating task τ_i to core $\mathcal{C}_j$ (lines 8-10). After adding τ_i, if the temporary set Γ' can be feasibly scheduled on core $\mathcal{C}_j$, the task is allocated to the core, and both Γ and Γ_j are updated (lines 11-12). The procedure then moves to the next iteration and considers the allocation of the next task. Otherwise, the task is not allocated and the procedure directly moves to the next iteration. If there is no feasible schedule under the constraints, the algorithm exits (lines 14-15). Otherwise, the algorithm outputs the task allocation scheme.

Algorithm 1: Determine Task-to-Core Allocation

```
Input: Task set $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ and core set $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$ satisfying $\mathcal{U}_1 \leq \mathcal{U}_2 \leq \cdots \leq \mathcal{U}_m$
Output: Task-to-core allocation scheme $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_m\}$
1  for $j = 1$ to $m$ do
2      initialize the task set allocated to core $\mathcal{C}_j$ by $\Gamma_j = \emptyset$ ;
3  $j = 1$;
4  while $\Gamma \neq \emptyset$ and $j \leq m$ do
5      for $i = 1$ to size($\Gamma$) do
6          calculate the vulnerability index $v_i$ of task $\tau_i$;
7      sort all tasks $\tau_i \in \Gamma$ in the decreasing order of task vulnerability
       index $v_i$ using Heapsort;
8      create a temporary set $\Gamma'$ to test the feasibility of task allocation;
9      for $i = 1$ to size($\Gamma$) do
           /* Group tasks in a first-fit manner */
10         $\Gamma' = \Gamma_j + \tau_i$;
11         if $\Gamma'$ can be feasibly scheduled on core $\mathcal{C}_j$ then
               /* Check deadline and schedulability */
12             $\Gamma = \Gamma - \tau_i$, $\Gamma_j = \Gamma_j + \tau_i$;
13     $j = j + 1$;
14 if $\Gamma \neq \emptyset$ and $j > m$ then
15     exit;
```

VII. Cross Entropy based Task Scheduling

Cross entropy (CE) method is a versatile heuristic tool that has been successfully applied to solve NP-hard combinatorial optimization problems [14], [15]. Our task frequency selection is an NP-hard combinatorial optimization problem. Thus, we apply the CE method to solve our problem. This section introduces the theoretic background of the CE method as well as the proposed CE-based task scheduling heuristic.

A. Theoretical Foundation of Cross Entropy

Consider an optimization problem with the objective of finding the minimum of real-valued function $\mathbb{F}(x)$ over the solution space $\mathcal{X}$, which is formulated as

$$S^* = \min\nolimits_{x \in \mathcal{X}} \mathbb{F}(x). \quad (10)$$

In CE method, the minimization objective associates with an estimation probability that is given by

$$\Psi(S) = \mathbb{P}_\sigma \Big(\mathbb{F}(X) \leq S \Big) = \mathbb{E}_\sigma [I_{\{\mathbb{F}(X) \leq S\}}] = \int_x I_{\{\mathbb{F}(X) \leq S\}} f(x; \sigma) \mathrm{d}x, \quad (11)$$

where S is a threshold, $\mathbb{P}_\sigma$ is the probability of $\mathbb{F}(X) \leq S$, $\mathbb{E}_\sigma$ is the expectation of $\mathbb{F}(X) \leq S$, and $I_{\{\mathbb{F}(X)\leq S\}}$ is the indicator function. $X = [X_1, X_2, \cdots, X_N]$ is a vector consisting of N random samples generated by a family of probability density functions (PDF) distributed in $\mathcal{X}$, denoted by $f(x;\omega)$ and parameterized by ω. In Eq. (11), ω in $f(x;\omega)$ is set to σ.

The target of CE optimization is to find the maximal S such that the probability of $\mathbb{F}(X) \leq S$, represented by $\Psi(S)$, approaches 0. At that moment, the probability of $\mathbb{F}(X) > S$ then approaches 1, indicating S is the maximum lower bound on the objective $\mathbb{F}(x)$ and thus the optimal solution. To realize the CE method, two key issues need to be solved. 1) How to calculate the $\Psi(S)$ for a given S? 2) How to derive a better S for a given current S?

1) In CE method, importance sampling technique is adopted to generate samples using an PDF $g(x)$ on $\mathcal{X}$ and compute the estimation of $\Psi(S)$ as $\widehat{\Psi}(S)$ using

$$\widehat{\Psi}(S) = \frac{1}{N}\sum\nolimits_{\ell=1}^{N} I_{\{\mathbb{F}(X_\ell)\leq S\}} \frac{f(X_\ell;\sigma)}{g(X_\ell)}. \qquad (12)$$

The optimal importance sampling PDF $g^*(x)$ for which the variance of $\widehat{\Psi}(S)$ is minimal, is expressed as

$$g^*(x) = I_{\{\mathbb{F}(X_\ell)\leq S\}} \cdot f(x;\sigma)/\Psi(S). \qquad (13)$$

The idea of CE method is to choose the importance sampling PDF g such that the Kullback-Leibler divergence (also the cross entropy) between the optimal importance sampling PDF g^* and g is minimal. The divergence is given by

$$\mathcal{D}(g^*, g) = \int g^*(x)\ln g^*(x)\mathrm{d}x - \int g^*(x)\ln g(x)\mathrm{d}x. \qquad (14)$$

Clearly, $\mathcal{D}(g^*, g)$ is minimal when $-\int g^*(x)\ln g(x)\mathrm{d}x$ is minimized, equivalent to finding the optimal δ to realize $\max_\delta \int g^*(x)\ln f(x;\delta)\mathrm{d}x$. The optimal δ, denoted by δ^*, is

$$\begin{aligned}\delta^* &= \arg\max_\delta \int g^*(x)\ln f(x;\delta)\mathrm{d}x \\ &= \arg\max_\delta \mathbb{E}_\sigma I_{\{\mathbb{F}(X_\ell)\leq S\}} \ln f(x;\delta). \end{aligned} \qquad (15)$$

2) The CE method iteratively derives a better S given the current S until the stop condition is met or the number of iterations reaches a predefined limit. Each solution is treated as a sample. In each round of iteration, samples are generated according to the PDF and their solution performance are evaluated. Afterwards, the best samples in terms of solution quality, called elite samples, are selected to update the characterizing parameter δ in the PDF. The updated PDF will be used to generate new samples in the next iteration.

B. Our CE-based Task Scheduling Heuristic

The objective of our task scheduling is to minimize SER degradation caused by satisfying system design constraints, which can be achieved by selecting the optimal frequencies for all tasks executing on individual cores. Due to the effectiveness of CE method in solving NP-hard combinatorial optimization problems, we apply it to solve our frequency selection problem. The basic idea of CE method has been introduced in Section VII-A. In our CE-based task scheduling heuristic, each sample represents a solution to the frequency setups of tasks executing on m cores, and the sample is deemed feasible when it meets the constraints given in Eqs. (4)-(6).

Algorithm 2: Determine Task Operating Frequency

```
Input: Task-to-core allocation scheme $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_m\}$ and core set
       $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$
Output: The frequency setup solution
1  initialize parameters of mean and variation with Gaussian distributions
   for all tasks allocated to $m$ cores: $\theta_1 = [\theta_1^{(1)}, \cdots, \theta_1^{(j)}, \cdots, \theta_1^{(m)}]$ and
   $\xi_1 = [\xi_1^{(1)}, \cdots, \xi_1^{(j)}, \cdots, \xi_1^{(m)}]$, where $\theta$ is the mean value, $\xi$ is the
   standard deviation, and $m$ is the number of cores;
2  $\eta = 1$;
3  repeat
4      generate $\mathcal{J}$ solution samples $[X_1, \cdots, X_\ell, \cdots, X_\mathcal{J}]$ using Latin
       hypercube sampling method according to distribution $N(\theta_\eta, \xi_\eta)$,
       where each sample $X_\ell = [X_\ell^{(1)}, \cdots, X_\ell^{(j)}, \cdots, X_\ell^{(m)}]$ is
       a solution to the frequency setup of tasks executing on $m$
       cores  /* $X_\ell^{(j)}$ represents the frequency setup of
       tasks in set $\Gamma_j$ allocated to core $\mathcal{C}_j$ */;
5      select $\mathcal{Q}$ ($\mathcal{Q} < \mathcal{J}$) feasible solution samples satisfying constraints
       in Eq. (4)-(6) from $\mathcal{J}$ samples using acceptance-rejection method;
6      evaluate the $\mathcal{Q}$ selected solution samples based on their system SER
       $R_{sys}$ calculated by Eq. (3);
7      choose top $\mathcal{K}$ ($\mathcal{K} < \mathcal{Q}$) elite samples with respect to $R_{sys}$ and define
       $\mathcal{I}$ as the set of indices of elite samples;
8      update the parameters $\theta_{\eta+1}$ and $\xi_{\eta+1}$: $\theta_{\eta+1}^{(j)} = \sum_{\ell\in\mathcal{I}} X_\ell^{(j)}/\mathcal{K}$ and
       $\xi_{\eta+1}^{(j)} = \sqrt{\sum_{\ell\in\mathcal{I}}(X_\ell^{(j)} - \theta_{\eta+1}^{(j)})/\mathcal{K}}$ ;
9      $\eta = \eta + 1$;
10 until predefined converge criteria of solutions is met or the total iteration
   number $\eta$ reaches the limit $\eta_{max}$;
```

The pseudo code of our heuristic is described in Algorithm 2. The algorithm takes as input task-to-core allocation scheme $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_m\}$ and core set $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m\}$. It first initializes the expected value and standard deviation of Gaussian distributions that are used to generate samples, and initializes the iteration counter η (lines 1-2). It then explores the optimal frequency setup solution in an iterative manner (lines 3-10). In each round of iteration, the algorithm generates $\mathcal{J}$ solution samples according to the distribution PDF and selects $\mathcal{Q}$ feasible samples that satisfy the design constraints using acceptance-rejection method (lines 4-5). Afterwards, the selected samples are evaluated in terms of system SER R_{sys} (line 6) and a certain number ($\mathcal{K}$) of elite samples are chosen to update the distribution PDF for generating a new set of solution samples used in the next iteration (lines 7-9). The iteration stops if the predefined converge criteria is satisfied or the total iteration number is reached. The algorithm can be strikingly accelerated by running it in the parallel programming environment. This is because that for each solution sample, the evaluation process of this solution sample is independent from other solution samples.

VIII. Evaluation

To evaluate the effectiveness of our scheme, we have performed several simulation-based studies. Specifically, we compare the system SER of our scheme to two representative methods: energy-aware reliability management (ERM) [16] and multi-objective optimization of reliability (MOR) [17]. ERM optimizes system overall energy consumption by determining the number of replicas and the frequency assignment for all tasks under a given SER target and all timing constraints. MOR develops a genetic algorithm based approach

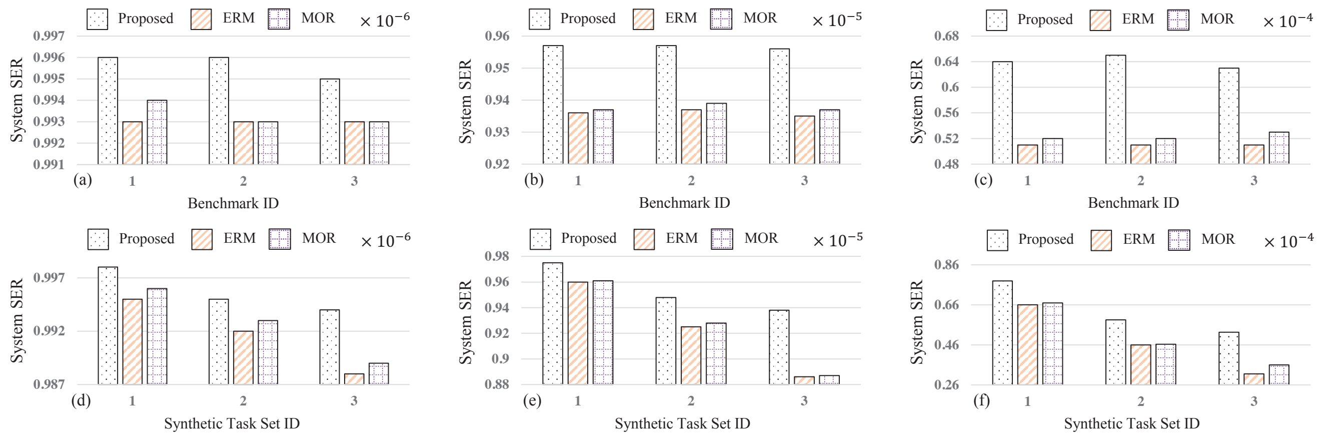

Fig. 4: System SER of (a)-(c) three real-world benchmarks and (d)-(f) three synthetic task sets under varying transient fault arrival rate levels ($\times 10^{-6}$, $\times 10^{-5}$, $\times 10^{-4}$) using the proposed scheme and representative methods ERM [16] and MOR [17].

to find the Pareto-optimization of SER and LTR by mapping tasks to cores and scaling core frequencies. We leave other performance evaluations to future work due to page limit.

Two sets of simulations have been carried out to validate our scheme. In the first set of simulations, three real-world benchmarks from Embedded System Synthesis Benchmark Suite [18] and the Nvidias TK1 board [19] that includes 4 high-performance cores and 1 low-power core, are utilized to verify the proposed algorithms. The three benchmarks are automotive, consumer, and telecom, which consist of 16, 13, 17 tasks, respectively. In the second set of simulations, three synthetic task sets are randomly generated to validate the proposed algorithms, which consist of 20, 40, 80 tasks, respectively. The three task sets are allocated to three simulated multicore systems, which include 5, 10, 20 cores, respectively. In the both sets of simulations, we investigate the system SER under varying transient fault rate levels (i.e., $\times 10^{-6}$, $\times 10^{-5}$, $\times 10^{-4}$) [1]. The running temperature is obtained using a widely used thermal modeling tool HotSpot [20] and the MTTF is derived using a system-level LTR modeling tool [7]. Details on parameter extraction are omitted due to page limit.

Fig. 4 (a)-(c) show the system SER of running three benchmarks on TK1 under different fault rate levels using our proposed scheme, ERM, and MOR. As can be readily seen, the system SER of our scheme is greater than that of ERM and MOR but not by too much when the fault rate is low (10^{-6}), whereas is far greater than that of ERM and MOR when the fault rate is high (10^{-4}). In the scenario of high fault rate, our scheme can improve the SER by up to 27.5% and 25% compared with ERM and MOR, respectively. Fig. 4 (d)-(f) show the system SER of running three synthetic task sets on simulated platforms under different fault rate levels using the proposed scheme, ERM, and MOR. We can find the same observation in these figures that our scheme outperforms ERM and MOR in terms of SER. The improvement of SER achieved by our scheme can be up to 66% and 45.3% when compared to ERM and MOR, respectively.

IX. Conclusion

We proposed a two-stage task allocation and scheduling scheme for real-time MPSoC systems to maximize soft-error reliability under the constraints of lifetime reliability, temperature, and deadline. Simulation results demonstrate that the proposed scheme can improve soft-error reliability by up to 66% as compared to representative methods ERM and MOR.

References

[1] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real- time embedded applications under hard energy constraint," *IEEE TII*, vol. 6, no. 3, pp. 316-328, 2010.

[2] G. Macario, M. Torchiano, and M. Violante, "An in-vehicle infotainment software architecture based on Google Android," *SIES*, pp. 257-260, 2009.

[3] M. Salehi, M. K. Tavana, S. Rehman, F. Kriebel, M. Shafique, A. Ejlali, and J. Henkel, "DRVS: Power-efficient reliability management through dynamic redundancy and voltage scaling under variations," *ISLPED*, pp. 225-230, 2015.

[4] J. Zhou and T. Wei, "Stochastic thermal-aware real-time task scheduling with considerations of soft errors," *Elsevier JSS*, vol. 102, pp. 123-133, 2015.

[5] Y. Ma, T. Chantem, R. P. Dick, and X. S. Hu, "Improving system-level lifetime reliability of multicore soft real-time systems," *IEEE TVLSI*, vol. 25, no. 6, pp. 1895-1905, 2017.

[6] T. Chantem, Y. Xiang, X. S. Hu, and R. P. Dick, "Enhancing multicore reliability through wear compensation in online assignment and scheduling," *DATE*, pp. 1373-1378, 2013.

[7] Y. Xiang, T. Chantem, R. P. Dick, X. S. Hu, and L. Shang, "System-level reliability modeling for MPSoCs," *CODES+ISSS* pp. 297-306, 2010.

[8] J. Zhou, X. S. Hu, Y. Ma, and T. Wei, "Balancing lifetime and soft-error reliability to improve system availability," *ASPDAC*, pp. 685-690, 2016.

[9] Y. Ma, T. Chantem, R. P. Dick, S. Wang, and X. S. Hu, "An on-line framework for improving reliability of real-time systems on Big-Little type MPSoCs," *DATE*, pp. 446-451, 2017.

[10] T. Kim, Z. Sun, H. Chen, H. Wang, and S. X. D. Tan, "Energy and lifetime optimizations for dark silicon many core microprocessor considering both hard and soft errors," *IEEE TVLSI*, vol. 25, no. 9, pp. 2561-2574, 2017.

[11] K. K. Rangan, M. Powell, G. Y. Wei, and D. Brooks, "Achieving uniform performance and maximizing throughput in the presence of heterogeneity," *HPCA*, pp. 3-14, 2011.

[12] M. Riera, R. Canal, J. Abella, and A. Gonzalez, "A detailed methodology to compute soft error rates in advanced technologies," *DATE*, pp. 212-222, 2016.

[13] X. Li, S. V, P. Bose, and J. River, "Architecture-level soft error analysis: examining the limits of common assumptions," *DSN*, pp. 266-275, 2007.

[14] X. Zhao, Y. Guo, Z. Feng, and S. Hu, "Parallel hierarchical cross entropy optimization for on-chip decap budgeting," *DAC*, pp. 843-848, 2010.

[15] R. Rubinstein and D. Kroese, "The cross entropy method: a unified approach to combinatorial optimization, monte-carlo simulation, and machine learning," *Annals of Operations Research*, 2004.

[16] M. A. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE TPDS*, vol. 28, no. 3, pp. 813-825, 2017.

[17] A. Das, A. Kumar, B. Veeravalli, C. Bolchini, and A. Miele, "Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs," *DATE*, pp. 1-6, 2014.

[18] E3S. [Online]. Available: http://ziyang.eecs.umich.edu/~dickrp/e3s/. 2013.

[19] Nvidia, "Jetson Tegra K1". [Online]. Available: https://developer.nvidia.com/embedded/develop/hardware.

[20] K. Skadron et al., "Temperature-aware microarchitecture: modeling and implementation," *ACM TACO*, vol. 1, no. 1, pp. 94-125, 2004.