Fault-Tolerant Mapping of Real-Time Parallel Applications under multiple DVFS schemes

Minyu Cui, Angeliki Kritikakou, Lei Mo, Emmanuel Casseau

Noname manuscript No.
(will be inserted by the editor)

Fault-Tolerant Mapping of Real-Time Parallel Applications under multiple DVFS schemes

Minyu Cui, Angeliki Kritikakou, Lei Mo, Emmanuel Casseau

Abstract On multicore platforms, reliable task execution, as well as low energy consumption, are essential. Dynamic Voltage/Frequency Scaling (DVFS) is typically used for energy saving, but with a negative impact on reliability, especially when the frequency is low. Using high frequencies to meet reliability constraints is not always feasible, while multiple replicas increase energy consumption. To minimize energy consumption, enhancing reliability, without violating real-time constraints, we propose an approach that combines distinct reliability enhancement techniques, under task-level, processor-level and system-level DVFS. Our task mapping problem jointly optimizes task allocation, task frequency assignment, and task duplication, under multiple constraints. This is achieved by formulating the task mapping problem as a mixed integer non-linear programming problem and equivalently transforming it into a mixed integer linear programming, that is optimally solved. From the obtained results, the proposed approach achieves better energy consumption and finds solutions, when other approaches fail.

Acknowledgements This work is founded by China Scholarship Council (CSC).

Minyu Cui
Univ Rennes, INRIA, CNRS, IRISA, France, E-mail: minyu.cui@irisa.fr

Angeliki Kritikakou
Univ Rennes, INRIA, CNRS, IRISA, France, E-mail: angeliki.kritikakou@irisa.fr

Lei Mo
School of Automation, Southeast University, China, E-mail: lmo@seu.edu.cn

Emmanuel Casseau
Univ Rennes, INRIA, CNRS, IRISA, France, E-mail: emmanuel.casseau@irisa.fr

1 Introduction

1.1 Context

Hard real-time applications require both *timing guarantees* and *reliable execution*, i.e., tasks must finish correctly and before their deadlines. To ensure timing guarantees, a safe estimation of the Worst-Case Execution Time (WCET) of tasks is typically used. Nevertheless, the correct execution of tasks is threatened by sources that affect hardware platforms, such as radiation and electromagnetic interference, causing non-persistent transient faults (also known as *soft errors*). As technology size reduces, platforms become more susceptible to soft errors. Meanwhile, the energy consumption of embedded systems has become a crucial factor, especially for systems with a limited energy budget. As a result, adaptive management techniques have been established to maximize energy efficiency [1]. DVFS is a technique that optimizes energy consumption by simultaneously scaling down the processor supply voltage and frequency, during execution. However, DVFS has a significant, negative impact on system reliability, mainly because of the increased transient fault rates at low supply voltage and frequency levels [2]. Therefore, task mapping techniques must take into account the reliability degradation, energy consumption and execution time in order to make provisions accordingly, especially for real-time applications [3].

1.2 Related work and motivation

Although both fault tolerance and energy management have been extensively (but often independently) studied, the co-management of system reliability and energy efficiency has been addressed only recently [4]. Table 1 depicts representative State-of-the-Art (SoA) task mapping approaches with DVFS, under Energy Budget (EB), Real-Time (RT) and Reliability (R) constraints, with maximizing system Reliability (mR) or minimizing Energy consumption (mE) goals. Fault tolerance is provided by Recovery tasks (Rec.) or task Replicas (Rep.). A task is executed successfully, if at least one replica is executed without faults [3]. Tasks are Independent (I) or Dependent (D). The platform has a Homogeneous (HO), Heterogeneous (HE) or Single (S) processor. Based on the problem formulation and solving method, the solutions are Feasible (F) or Optimal (O).

Regarding reliability maximization, approaches without fault tolerance map only original tasks, e.g., through online heuristics [5]. Approaches with fault-tolerance i) insert a recovery task, always executed with maximum frequency in case of an error [2], ii) map a given number of task replicas [6], and iii) decide the number of replicas during task mapping [7].

Regarding minimizing energy consumption, approaches exist without fault tolerance, e.g., by decomposing the task mapping problem into two sub-problems (satisfying reliability goal and minimizing resource consumption) [8]. To tackle

the negative impacts of DVFS on reliability, existing techniques preserve the original system reliability for all tasks, i.e., the reliability obtained with the maximum platform frequency. Slack is reserved to execute a recovery task (individual [9] or shared [4]) with the maximum frequency. Then, DVFS scales down tasks. If an error is detected, the recovery task is evoked. A longest-task-first heuristic applies DVFS and recovery tasks only to a task subset [10]. However, such approaches usually require high frequencies to satisfy the original reliability, increasing energy consumption. It is also possible that reliability constraints cannot be satisfied, even with the maximum frequency, leading to empty solution space.

Other approaches compute the required number of replicas to always meet reliability constraints. Heuristics are typically proposed, due to high complexity, e.g., a first-fit decreasing heuristic for homogeneous platforms [3] and a list scheduling heuristic for heterogeneous platforms [1]. Hybrid approaches use a given number of replicas and apply task recovery, when an error occurs [13]. However, the increased number of replicas leads to large energy consumption, combined with a negative impact on execution time. When the real-time constraints are strict, solutions may not exist. To reduce this negative impact, the number of replicas is restricted. In [12], a heuristic decides among single execution, task duplication and task triplication. However, no guarantees are provided on real-time and reliability constraints. In [15], a heuristic determines which tasks to be duplicated, removing the need of a recovery task, without considering any reliability constraint. An Integer Linear Programming (ILP) approach [14] maps independent tasks on a heterogeneous platform to satisfy a given percentage of duplicated tasks, under energy budget constraint. However, the reliability of tasks is not considered.

Table 1: Comparison with representative State-of-the-Art

Ref.	Goal		Fault tol.		Tasks		Platform			Constraints			Sol.	
	mE	mR	Rec.	Rep.	I	D	HO	HE	S	RT	EB	R	F	O
[5]		√				√		√			√		√	
[6]		√		√		√	√			√	√		√	
[2]		√	√			√			√	√	√			√
[7]		√		√		√		√				√	√	
[8]	√					√		√				√	√	
[9, 10]	√		√			√		√		√		√	√	
[4]	√		√			√			√	√		√	√	
[11]	√		(√)			√		√		√		√	√	
[3]	√			√	√		√			√		√	√	
[1]	√			√		√		√				√	√	
[12]	√			√		√	√			(√)		(√)	√	
[13]	√			√		√	√			√		√	√	
[14]	√			√	√			√				(√)		√
[15]	√		√	√		√	√			√			√	
Prop.	√			√		√	√			√		√		√

1.3 Contributions

This work belongs to the last category, extending the SoA by proposing an optimal task mapping approach, which decides between reliable single task execution and task duplication, under real-time and reliability constraints. The execution of a task is considered safe, when it meets its reliability constraint. Otherwise, the task requires duplication. As a result, the number of duplicated tasks is reduced, relieving the negative impact on execution time and energy consumption. We focus on the design-time phase, where task mapping and scheduling is performed offline. The goal is to minimize energy consumption, by optimally and concurrently deciding which tasks to be duplicated, the frequency per task and the task allocation, under both real-time and reliability constraints. Unlike the majority of SoA techniques, original and duplicated tasks can have different operating frequencies, being more suitable for real-world systems [6]. To achieve that, the task mapping problem is formulated as a Mixed Integer Non-Linear Programming (MINLP), and safely transformed into a Mixed Integer Linear Programming (MILP) problem, in order to be solved optimally. Last, but not least, the proposed approach is extended for three DVFS schemes, typically supported by hardware platforms. To support our contribution, we performed extensive experiments, under different scenarios. From the results, the proposed approach can find feasible solutions, when existing replication approaches fail, and consumes less energy, compared to existing reliability-aware approaches.

The rest of the paper is organized as follows. Section 2 introduces the system model. Section 3 presents the formulation of the proposed approaches. Section 4 presents the evaluation results. Finally, Section 5 concludes this study.

2 System Model

2.1 Task Model

This work considers a real-time application consisting of N frame-based non-preemptive dependent tasks $\{\tau_1, \ldots, \tau_N\}$. The application is represented by a Directed Acyclic Graph (DAG) $G(\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ denotes the set of tasks and $\mathbb{E}$ represents the partial order, corresponding to the precedence constraints among tasks. All the tasks are released at time 0. The ready time of a task is the time instant at which all its predecessors have been completed. Each task τ_i is defined by its deadline D_i and Worst-Case Execution Cycles (WCEC) C_i. The common period of the tasks is given by the application frame, denoted by H. Hence, the deadline D_i cannot be larger than the application frame H, i.e., $D_i \leq H$. Different functions within an application exhibit distinct vulnerabilities to soft errors, due to variations in the spatial and temporal vulnerabilities of different instructions [12]. Hence, each task τ_i has its own reliability threshold R_i^{th}.

2.2 Platform Model

A shared-memory multicore platform is used with M homogeneous processors $\{\theta_1, \ldots, \theta_M\}$. DVFS is possible at the task (TL-DVFS), processor (PL-DVFS) and system levels (SL-DVFS). Such variants correspond to DVFS-schemes in existing hardware platforms, e.g., clock frequency is controlled per core in Intel-Xeon E5620 and per 8 cores in GreenWaves GAP8. There are L pairs of frequency/voltage levels $\{(f_1, v_1), \ldots, (f_L, v_L)\}$. Similar to [1, 4, 13], the relationship of voltage and frequency is almost linear. Therefore, we use the term frequency scaling to express the simultaneous change of voltage and frequency. When task τ_i is executed with frequency f_l, its execution time is $t_i = \frac{C_i}{f_l}$. The power consumption is modeled as the sum of static power P_l^{sta} and dynamic power P_l^{dyn} with the given voltage/frequency level (v_l, f_l), i.e., $P_l = P_l^{sta} + P_l^{dyn} = P_l^{sta} + C_{eff} v_l^2 f_l$, where C_{eff} is the effective switching capacitance [16].

2.3 Processor Fault Model and Reliability

As transient faults have a higher occurrence than permanent faults [8], we focus on transient faults and fault detection is assumed to be done by the hardware. Our fault model is the one typically used by the SoA, such as [1, 4, 7, 8, 13], i.e., a Poisson distribution with an average fault rate $\lambda(f)$ at frequency f. Different frequency scalings yield various failure rates. The failure rate per time unit of a processor at frequency f is given by $\lambda(f) = \lambda_0 \times 10^{d_0 \frac{f_{\max} - f}{f_{\max} - f_{\min}}}$, where $f_{\max} = \max_{\forall l}\{f_l\}$, $f_{\min} = \min_{\forall l}\{f_l\}$, λ_0 is the average failure rate of the processor corresponding to $f_{\max}$, and d_0 is a positive constant, indicating the sensitivity of failure rates to voltage scaling [1]. The reliability of a task τ_i is defined as the probability of executing a task without any fault, based on the exponential model of [17]. Therefore, the reliability of a task with WCEC C_i at frequency f_l is $r_i(f_l) = e^{-\varphi_i(f_l)}$, where $\varphi_i(f_l) = \lambda(f_l) \times t_i$, and $t_i = \frac{C_i}{f_l}$ is the execution time of task τ_i at frequency f_l [2, 3, 17]. If the reliability of task τ_i is larger than its reliability threshold R_i^{th}, the execution is considered to be reliable [8] and the task reliability is not modified, i.e., $R_i = r_i(f_l)$. Otherwise, the task τ_i is duplicated and the duplicated task is executed on a different processor, since it is unlikely that the execution of both original and duplicated tasks on different processors fails [3, 15]. If the duplicated task τ_{N+i} (task index $N + i$ is defined in Section 3) is executed with frequency f_l', its reliability is $r_{N+i}(f_l')$. Therefore, the total reliability is $R_i = 1 - [1 - r_i(f_l)][1 - r_{N+i}(f_{l'})]$. As our approach finds an offline optimal task mapping solution, with the goal of minimizing energy consumption, we consider only task duplication, i.e., a single replica per task, in order not to unnecessarily increase the number of replicas, in case no faults occur. An online mechanism can be applied to further improve the energy consumption of our solution (by not executing the duplicated task, when the first execution is

correct), and to deal with the low probability cases, where both original and duplicated tasks are faulty. In this work, cores are assumed to operate below a temperature threshold, where temperature impact on reliability is low [18].

3 Reliability-aware Fault-tolerant Task Mapping Approach

The proposed *Reliability-aware Fault-tolerant Task Mapping (RAFTM)* approach minimizes the system energy consumption subject to a set of reliability and real-time constraints. It concurrently decides: 1) task frequency assignment ($\boldsymbol{s}$), 2) task duplication decision ($\boldsymbol{\sigma}$), 3) task allocation ($\boldsymbol{q}$), and 4) task start time ($\boldsymbol{t^s}$). Note that the task set is extended with N duplicated tasks, i.e., $\overline{\mathcal{N}} = \{1, \ldots, N, N+1, \ldots, 2N\}$. Tasks $\{\tau_1, \ldots, \tau_N\}$ are original tasks and tasks $\{\tau_{N+1}, \ldots, \tau_{2N}\}$ are duplicated tasks. Hence, task τ_{N+1} is the duplicated task of τ_1. Duplicated tasks have the same WCEC and deadline as the original ones. Although initially all tasks are considered duplicated, the approach decides whether the duplication is maintained, through variables $\sigma_{\mathbf{i}}$, according to the real-time and reliability constraints. A task waits for both original and duplicated tasks of its predecessors to end, before starting execution [1]. The proposed approach's formulation is initially presented for a multicore platform supporting the TL-DVFS scheme. Then, it is adapted for PL-DVFS and SL-DVFS schemes. Table 2 summarizes the notation.

Parameters	
$\mathcal{M}$	$\{1, \ldots, M\}$, with M number of processors
$\mathcal{N}$	$\{1, \ldots, N\}$, with N number of tasks
$\overline{\mathcal{N}}$	$\{1, \ldots, 2N\}$, with $2N$ number of tasks
$\mathcal{L}$	$\{1, \ldots, L\}$, with L number of voltage/frequency levels
C_i	WCEC of task τ_i
D_i	deadline of task τ_i
(v_l, f_l)	the l^{th} voltage/frequency level
R_i^{th}	reliability threshold of task τ_i
O_{ij}	$O_{ij} = 1$ if task τ_j is dependent on task τ_i, else, $O_{ij} = 0$
Binary Variables	
$\sigma_i = 1$	if task τ_i is executed, else 0
$q_{im} = 1$	if task τ_i executes on processor θ_m, else 0
$w_{ij} = 1$	if task τ_i executes before task τ_j on same processor, else 0
TL-DVFS:	$s_{il} = 1$, if task τ_i executes with f_l, else 0
PL-DVFS:	$s_{ml} = 1$, if processor θ_m executes with f_l, else 0
SL-DVFS:	$s_l = 1$, if the system executes with f_l, else 0
Continuous Variables	
t_i^s	the start time of task τ_i

Table 2: Main Notations

3.1 Task-Level DVFS scheme (TL-DVFS)

The following paragraphs describe the constraints and the objective function of the task mapping TL-DVFS problem and its equivalent transformation.

3.1.1 Frequency assignment

Each task has a single frequency:

$$\sum_{l \in \mathcal{L}} s_{il} = \sigma_i, \ \forall i \in \overline{\mathcal{N}}. \tag{1}$$

3.1.2 Reliability and Task duplication decision

For an original task τ_i ($1 \leq i \leq N$), $\sigma_i = 1$. Its reliability is expressed as $r_i = \sum_{l \in \mathcal{L}} s_{il} e^{-\varphi_i(f_l)}$, where $\varphi_i(f_l) = \lambda(f_l) \times \frac{C_i}{f_l}$ (Section 2.3). If $r_i > R_i^{th}$, task τ_i does not need duplication ($\sigma_{N+i} = 0$). Else, if $0 < r_i \leq R_i^{th}$, the task τ_i is duplicated, i.e., $\sigma_{N+i} = 1$. To transfer this "if-else" constraints into linear mathematical formula, we use the following Lemma:

Lemma 1 *Let x and y denote two discrete variables where $0 < x_{\min} \leq x \leq x_{\max} \leq 1$ and $0 < y_{\min} \leq y \leq y_{\max} \leq 1$. Let c denote a binary variable. Given the determination i) if $0 < x \leq y$, $c = 1$, and ii) if $x > y$, $c = 0$, then the determination can be equivalently transferred into $\delta - (1+\delta)c \leq x - y \leq 1 - c$, where δ is a positive small value.*

Proof To prove that the given determination can be equivalently transferred into the linear constraint, we first let $C_1 : \delta - (1+\delta)c \leq x - y$ and $C_2 : x - y \leq 1 - c$. i) If $x < y$, then $x - y < 0$. For C_1, c must be 1. For C_2, c can be either 0 or 1. To satisfy both C_1 and C_2, we have $c = 1$. If $x = y$, for C_1, c must be 1 due to $x - y = 0$ and $\delta > 0$. For C_2, c can be either 0 or 1. Similarly, we obtain $c = 1$. ii) If $x > y$, for C_1, c can be either 0 or 1. However, c must be 0 in C_2 due to $x - y > 0$. Hence, c must be 0 if $x > y$.

Since there are L pairs of voltage/frequency, we have $r_i \in \{e^{-\varphi_i(f_1)}, \ldots, e^{-\varphi_i(f_L)}\}$. On this basis, the relationship between R_i, r_i^{th} and σ_i is linearized as:

$$\delta_i - (1+\delta_i)\sigma_{N+i} \leq \sum_{l \in \boldsymbol{L}} s_{il} e^{-\varphi_i(f_l)} - R_i^{th} \leq 1 - \sigma_{N+i}, \ \forall i \in \mathcal{N}. \tag{2}$$

Thus, the reliability of task τ_i becomes $R_i = (1 - \sigma_{N+i}) r_i + \sigma_{N+i}[1 - (1 - r_i)(1 - r_{N+i})]$, where r_i is the reliability of original τ_i and r_{N+i} is the reliability of duplication task τ_{N+i}, if duplication takes place.

However, if the reliability after duplication does not satisfy the reliability threshold, then the solution is not acceptable, so we have to guarantee that the reliability of a task considering duplication should not be smaller than its reliability threshold

$$R_i \geq R_i^{th}, \ \forall i \in \mathcal{N}. \tag{3}$$

3.1.3 Task allocation

A task τ_i is executed on one processor:

$$\sum_{m \in \mathcal{M}} q_{im} = \sigma_i, \ \forall i \in \overline{\mathcal{N}}. \tag{4}$$

If task τ_i is duplicated, original and duplicated tasks should be allocated on different processors. We have:

$$q_{im} + q_{N+i,m} \leq 1, \ \forall i \in \mathcal{N}, \ \forall m \in \mathcal{M}. \tag{5}$$

3.1.4 Task non-overlapping

When task τ_i is executed on processor θ_m with frequency f_l, its execution time is $\sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l}$ and, as tasks are executed in a non-preemptive manner, its end time is $t_i^e = t_i^s + \sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l}$. We need to guarantee that a task execution cannot overlap with any other task execution, when assigned to the same processor. The ordering of tasks, assigned on the same processor, is given by a matrix $\boldsymbol{w} = [w_{ij}]_{2N \times 2N}$. For any two tasks τ_i and τ_j, when $w_{ij} = 1$, tasks τ_i and τ_j are allocated on the same processor, and τ_i is executed before τ_j. Otherwise, either τ_i and τ_j are allocated on different processors or τ_i is executed after τ_j. Therefore, when both $q_{im} = 1$ and $q_{jm} = 1$, only one of w_{ij} and w_{ji} can be equal to 1, i.e., $w_{ij} + w_{ji} = 1$. Otherwise, both w_{ij} and w_{ji} are equal to 0, i.e., $w_{ij} + w_{ji} = 0$. On this basis, the non-overlapping constraints are formulated as:

$$t_i^e \leq t_j^s + (2 - q_{im} - q_{jm})H + (1 - w_{ij})H, \quad \forall i, j \in \overline{\mathcal{N}}, \ \forall m \in \mathcal{M}, \ i \neq j, \tag{6}$$

$$w_{ij} + w_{ji} = \sum_{m \in \mathcal{M}} q_{im} q_{jm}, \ \forall i, j \in \overline{\mathcal{N}}, \ i \neq j, \tag{7}$$

where $H = \max_{\forall i \in \mathcal{N}} \{D_i\}$.

3.1.5 Task dependencies

Based on the task graph, we introduce a Task Dependency matrix $\boldsymbol{o} = [o_{ij}]_{2N \times 2N}$. When $o_{ij} = 1$, task τ_i precedes τ_j and (8) becomes $t_j^s \geq t_i^s + \sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l}$. Otherwise, (8) is always satisfied.

$$t_j^s + (1 - o_{ij})H \geq t_i^s + o_{ij} \sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l}, \ \forall i, j \in \overline{\mathcal{N}}, \ i \neq j. \tag{8}$$

3.1.6 Real-time execution

Since all tasks should be executed before their deadline D_i, for each task:

$$t_i^s + \sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l} \leq D_i, \ \forall i \in \overline{\mathcal{N}}. \tag{9}$$

3.1.7 Objective Function

The system energy consumption is:

$$E_s = \sum_{i \in \overline{\mathcal{N}}} \left(\sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l} P_l \right), \tag{10}$$

Thus, the Primal Problem (**PP**) is formulated as an MINLP:

$$\mathbf{PP}: \min_{\substack{\boldsymbol{s},\boldsymbol{q},\boldsymbol{\sigma},\\ \boldsymbol{w},\boldsymbol{t^s}}} \sum_{i \in \overline{\mathcal{N}}} \left(\sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l} P_l \right)$$
$$\text{s.t.} \begin{cases} (1) \sim (9) \\ s_{il}, q_{im}, \sigma_i, w_{ij} \in \{0,1\}, \ 0 \leq t_i^s \leq D_i, \\ \forall i \in \overline{\mathcal{N}}, \ \forall m \in \mathcal{M}, \ \forall l \in \mathcal{L}. \end{cases}$$

In order to find the optimal solution, we simplify the problem structure by transforming **PP** to an equivalent MILP, through variable replacement, which removes the nonlinear variable combinations based on the following lemma:

Lemma 2 *Let b_1, b_2 and g denote binary variables. The nonlinear item $g = b_1 b_2$ can be replaced by i) $g \leq b_1$, ii) $g \leq b_2$ and iii) $g \geq b_1 + b_2 - 1$.*

Proof Inequalities $g \leq b_1$ and $g \leq b_2$ ensure that g is zero, if either $g \leq b_1$ or $g \leq$ is zero. Equality $g \geq b_1 + b_2 - 1$ ensures that g becomes 1, if both variables b_1 and b_2 are 1.

Let $h_{ij}^m = q_{im} q_{jm}$ $(i, j \in \overline{\mathcal{N}}, m \in \mathcal{M}, i \neq j)$ and $h_{ii}^m = 0$, when $i = j$. (7) can be equally expressed as

$$w_{ij} + w_{ji} = h_{ij}^m, \ \forall i, j \in \overline{\mathcal{N}}, \ i \neq j, \tag{11}$$

$$-q_{im} + h_{ij}^m \leq 0, \ -q_{jm} + h_{ij}^m \leq 0, \ q_{im} + q_{jm} - h_{ij}^m \leq 1,$$
$$\forall i, j \in \overline{\mathcal{N}}, \ \forall m \in \mathcal{M}, \ i \neq j. \tag{12}$$

Then, **PP** is reformulated as the following MILP problem:

$$\mathbf{RAFTM\text{-}TL}: \min_{\substack{\boldsymbol{s},\boldsymbol{q},\boldsymbol{\sigma},\\ \boldsymbol{w},\boldsymbol{h},\boldsymbol{t^s}}} \sum_{i \in \overline{\mathcal{N}}} \left(\sum_{l \in \mathcal{L}} s_{il} \frac{C_i}{f_l} \right) P_l$$
$$\text{s.t.} \begin{cases} (1) \sim (6), (8) \sim (9), (11) \sim (12), \\ s_{il}, q_{im}, \sigma_i, w_{ij}, h_{ij}^m \in \{0,1\}, 0 \leq t_i^s \leq D_i, \\ \forall i \in \overline{\mathcal{N}}, \ \forall m \in \mathcal{M}, \ \forall l \in \mathcal{L}. \end{cases}$$

3.2 Processor-Level DVFS scheme (PL-DVFS)

For platforms supporting PL-DVFS, each processor can have a single frequency level:

$$\sum_{l\in\mathcal{L}} s_{ml} = 1,\ \forall m \in \mathcal{M}, \tag{13}$$

For a task τ_i, the frequency is $\sum_{m\in\mathcal{M}} q_{im}\left(\sum_{l\in\mathcal{L}} s_{ml}\right) f_l$, the reliability is $R_i = \sum_{m\in\mathcal{M}} q_{im}\left(\sum_{l\in\mathcal{L}} s_{ml}\right) e^{-\varphi_i(f_l)}$ and the execution time is $\sum_{m\in\mathcal{M}} q_{im}\left(\sum_{l\in\mathcal{L}} s_{ml}\right)\frac{C_i}{f_l}$. These equations replace the corresponding equations of RAFTM-TL, and the proposed approach with PL-DVFS is formulated as

$$\textbf{RAFTM-PL}: \min_{\substack{\boldsymbol{s},\boldsymbol{q},\boldsymbol{\sigma},\\ \boldsymbol{w},\boldsymbol{t^s}}} \sum_{i\in\overline{\mathcal{N}}}\left[\sum_{m\in\mathcal{M}} q_{im}\left(\sum_{l\in\mathcal{L}} s_{ml}\frac{C_i}{f_l}P_l\right)\right]$$
$$\text{s.t.}\begin{cases}(2)\sim(9),(13)\\ s_{ml}, q_{im}, \sigma_i, w_{ij}\in\{0,1\},\ 0\le t_i^s\le D_i,\\ \forall i\in\overline{\mathcal{N}},\ \forall m\in\mathcal{M},\ \forall l\in\mathcal{L}.\end{cases}$$

3.3 System-Level DVFS scheme (SL-DVFS)

For platforms supporting SL-DVFS, all processors are assigned with same frequency:

$$\sum_{l\in\mathcal{L}} s_l = 1. \tag{14}$$

For a task τ_i, the frequency is $\sum_{l\in\mathcal{L}} s_l f_l$, the reliability is $R_i = \sum_{l\in\mathcal{L}} s_l e^{-\varphi_i(f_l)}$ and the execution time is $\sum_{l\in\mathcal{L}} s_l\frac{C_i}{f_l}$. Then, the proposed approach with SL-DVFS is formulated as

$$\textbf{RAFTM-SL}: \min_{\substack{\boldsymbol{s},\boldsymbol{q},\boldsymbol{\sigma},\\ \boldsymbol{w},\boldsymbol{t^s}}} \sum_{i\in\overline{\mathcal{N}}}\left(\sum_{l\in\mathcal{L}} s_l\frac{C_i}{f_l}P_l\right)$$
$$\text{s.t.}\begin{cases}(2)\sim(9),(14)\\ s_l, q_{im}, \sigma_i, w_{ij}\in\{0,1\},\ 0\le t_i^s\le D_i,\\ \forall i\in\overline{\mathcal{N}},\ \forall m\in\mathcal{M},\ \forall l\in\mathcal{L}.\end{cases}$$

4 Experimental Results

4.1 Experimental set-up

The processor model is a RISC-V Instruction Set Architecture (ISA), i.e., the open-source Comet processor, which is a high-level C++ model 32-bit RISC-V ISA and standard 5-stage pipeline [19]. Regarding the DVFS, $L = 6$

voltage/frequency levels are used [20] considering 64 nm technology, as depicted in Table 3. To obtain realistic inputs for our experiments, we count the execution cycles and Memory Accesses (MA) of common benchmarks from MiBench suite [21], using Comet simulator. The sources of timing variability are eliminated to obtain safe and context-independent measurement [22] without interferences ($WCEC_{iso}$). Then, the $WCEC_{inf}$, considering worst case interferences from the other cores, is computed. As the contribution of this paper is not WCET estimation, a trivial pessimistic approach is applied: all cores may conflict during a memory access. Thus, the interference cost is given by (M-1)*MA*Main_Memory_Access_Delay (Table 3).

The proposed Reliability-Aware Fault-Tolerant Mapping (RAFTM) approach is compared to two SoA approaches: i) a Reliability-Aware Mapping (RAM) approach, which chooses the frequency to satisfy the reliability constraint for each task without task replication, similar to [1, 8], and ii) a Task Duplication Mapping (TDM) approach, which always duplicates every task, similar to [1] and [3] (when the replica number of each task is equal to two) and to [14] (when the percentage of duplicated task is set to 100%). RAM is the common way to meet the expected reliability without taking replication into account. Considering task replication, TDM is the least energy consuming approach, compared to other methods that use more than two replicas per task. A large and diverse set of experiments is performed, by tuning the:

1. Number of processors ($M = 2, 4, 6$).
2. Platform DVFS scheme (TL-DVFS, PL-DVFS and SL-DVFS).
3. Average failure rate ($\lambda_0 = 5 \times 10^{-5}$, 4×10^{-4}, 5×10^{-4} faults/sec) with a failure rate constant ($d_0 = 3$) of processor fault model [17].
4. Size of task set ($\overline{\mathcal{N}} = 10, 20, 30$).
5. For a task set size, 10 experiments are performed. For each experiment, the characteristics of a task are:
   – WCEC: Based on $WCEC_{inf}$ (Table 3), the WCEC of each task is selected within the range $[1 \times 10^8, 4 \times 10^8]$, incorporating the time overhead for

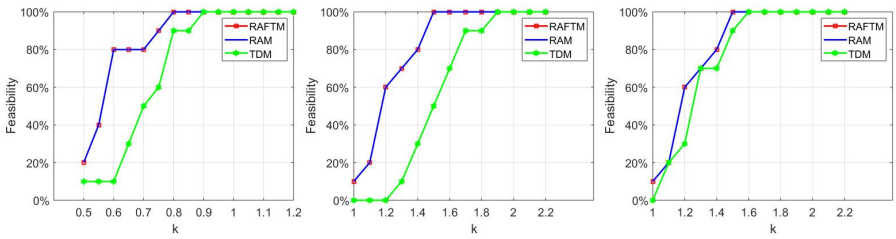
Table 3: Platform and benchmark characteristics

v_l (V)	0.85	0.90	0.95	1.00	1.05	1.1
f_l (GHz)	0.801	0.8291	0.8553	0.8797	0.9027	1.0
C_{eff}	7.3249	8.6126	10.238	12.315	14.998	18.497
Main Memory Access Delay					200 cycles	

Benchmark	**MA**	$WCEC_{iso}$	$WCEC_{inf}$		
			$M = 2$	$M = 4$	$M = 6$
matmul_int	371,957	3,313,958	77,705,358	226,488,158	449,662,358
matmul_int64	507,133	4,055,289	78,446,689	308,335,089	612,614,889
qsort_int	184,089	875,616	75,267,016	111,329,016	221,782,416
qsort_int64	259,553	1,219,854	75,611,254	156,951,654	312,683,454
qsort_float	185,437	1,745,122	76,136,522	113,007,322	224,269,522
dijkstra	117,151	766,369	75,157,769	71,056,969	141,347,569
blowfish	110,330	3,058,991	77,450,391	69,256,991	135,454,991
stringsearch	597,608	13,093,544	87,484,944	371,658,344	730,223,144

frequency (and supply voltage) changes (e.g., 10–150 μs [9, 10]) and the sanity checks at the end of a task [3].
- Reliability threshold R_i^{th}: Selected within the range $[0.9990, 0.9999]$, considering a typical magnitude 10^{-3} for reliability target [3]. Such a reliability target for a task is inline with safety standards, such as ISO26,262 for automotive systems, DO-178B for avionics systems and IEC 61,508 for industrial software systems [1] [8].
- Deadline: Tuned from strict deadlines to relaxed ones, by changing k in $D_i = \hat{t_i^s} + k * d_i$, where $\hat{t_i^s}$ is the relative start time of task τ_i equal to $\max_{\forall j \in Pre\{i\}}\{D_j\}$ (with $Pre\{i\}$ the predecessors of task τ_i) and the relative deadline given by $d_i \in \left[\frac{C_i}{f_{\max}}, \frac{C_i}{f_{\min}}\right]$.

Note that the above numbers provide only specific values to problem parameters, without affecting the problem structure. The approaches are implemented and solved with Gurobi 9.0.1 (MILP solver) on several servers, as hundreds of experiments took place. Due to page limitations, we present a subset of the experiments (TL-DVFS, PL-DVFS and SL-DVFS for $\overline{\mathcal{N}} = 10, M = 4$, $\overline{\mathcal{N}} = 20, M = 4$, and $\overline{\mathcal{N}} = 20, M = 6$) and we comment on the rest of the experiments.



(a) $\overline{\mathcal{N}} = 10, M = 4$ (b) $\overline{\mathcal{N}} = 20, M = 4$ (c) $\overline{\mathcal{N}} = 20, M = 6$

Fig. 1: Feasibility for a) $\overline{\mathcal{N}} = 10, M = 4$, b) $\overline{\mathcal{N}} = 20, M = 4$, and c) $\overline{\mathcal{N}} = 20, M = 6$ for all DVFS schemes.

To evaluate the approaches, the following metrics are used:

1. Feasibility, i.e., percentage of experiments that found the optimal solution in these 10 experiments.
2. Energy consumption of RAM and TDM, compared to the energy consumption of the proposed RAFTM approach.
3. Reliability Improvement ($RI = R_i - R_i^{th}$), i.e., the task reliability *above* the reliability threshold.
4. Task duplication, i.e., the percentage of tasks that RAFTM decided to duplicate.
5. Time, i.e., the time required for each approach to find a solution.

To better explain the performance, we organize our experiments in two set-ups of reliability threshold:

1. Set-up 1: The task reliability threshold can always be met by executing only the original task using a high processor frequency. To achieve that, the reliability threshold is selected within $[0.9990, 0.9995]$ and the average failure rate λ_0 equal to 5×10^{-5} faults/sec. In this set-up, the feasibility of RAFTM and RAM are the same.
2. Set-up 2: The task reliability threshold cannot always be met by executing only the original task, even at maximum frequency. To achieve that, the range of reliability constraint is increased, i.e., $[0.9990, 0.9999]$. We explore the behavior of the proposed RAFTM and RAM approaches with the average failure rate λ_0 is equal to 5×10^{-5}, 4×10^{-4}, and 5×10^{-4} faults/sec.

4.2 Experimental set-up 1

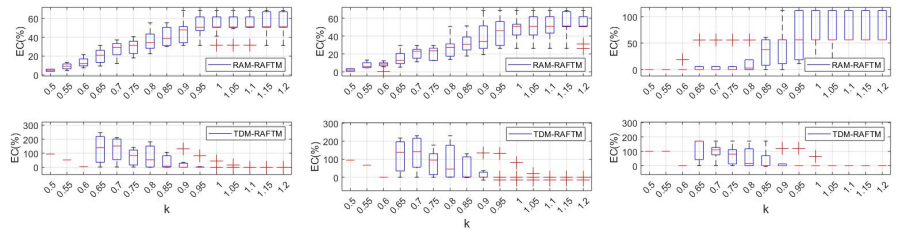
(a) $\overline{\mathcal{N}} = 10$, $M = 4$ (b) $\overline{\mathcal{N}} = 20$, $M = 4$ (c) $\overline{\mathcal{N}} = 20$, $M = 6$

Fig. 2: RAFTM duplication percentage: a) $\overline{\mathcal{N}} = 10$, $M = 4$, b) $\overline{\mathcal{N}} = 20$, $M = 4$, and c) $\overline{\mathcal{N}} = 20$, $M = 6$ (all DVFS schemes).

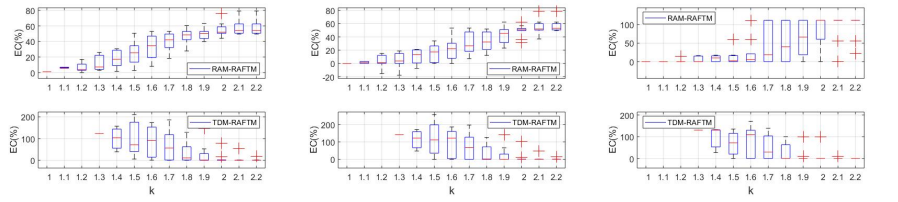
4.2.1 Feasibility

The feasibility of RAFTM, RAM and TDM is depicted in Fig. 1, for $\overline{\mathcal{N}} = 10$ and $\overline{\mathcal{N}} = 20$ with $M = 4$, and for $\overline{\mathcal{N}} = 20$ with $M = 6$.

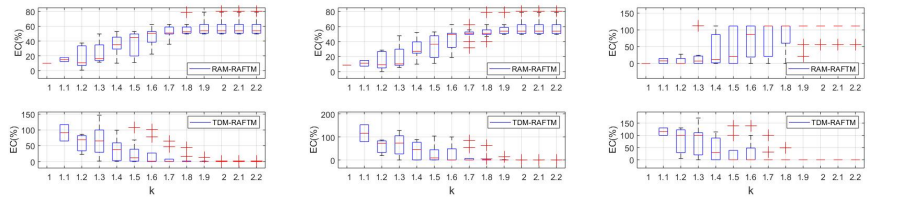
Regarding the comparison with TDM, RAFTM is able to find solutions in significantly more experiments than TDM, since RAFTM is not obliged to duplicate reliable tasks, compared to TDM. When feasibility is not 100% for both approaches, RAFTM can find a solution, on average, in 30%, 33% and 10% more experiments, than TDM (Fig. 1). When $k = 1, 1.1, 1.2$ ($\overline{\mathcal{N}} = 20$ and $M = 4$), TDM cannot find any solution. As the number of processors is increased, e.g., from $M = 4$ to $M = 6$, the capability of TDM to find solutions improves, as more cores are available to schedule the duplicated tasks. Note that RAFTM achieves 100% feasibility in earlier deadlines than TDM, i.e., $k = 0.8$ ($\overline{\mathcal{N}} = 10$ and $M = 4$) and $k = 1.5$ ($\overline{\mathcal{N}} = 20$ and $M = 4$, $\overline{\mathcal{N}} = 20$ and $M = 6$). When the deadline becomes relaxed (and TDM can *always* find solutions), i.e., $k = 0.9$ ($\overline{\mathcal{N}} = 10$ and $M = 4$), $k = 1.9$ ($\overline{\mathcal{N}} = 20$ and $M = 4$),



(a) TL-DVFS ($\overline{\mathcal{N}} = 10$, $M = 4$) (b) PL-DVFS ($\overline{\mathcal{N}} = 10$, $M = 4$) (c) SL-DVFS ($\overline{\mathcal{N}} = 10$, $M = 4$)



(d) TL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 4$) (e) PL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 4$) (f) SL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 4$)

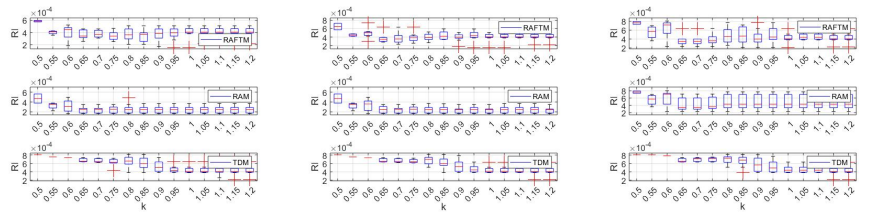


(g) TL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 6$) (h) PL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 6$) (i) SL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 6$)

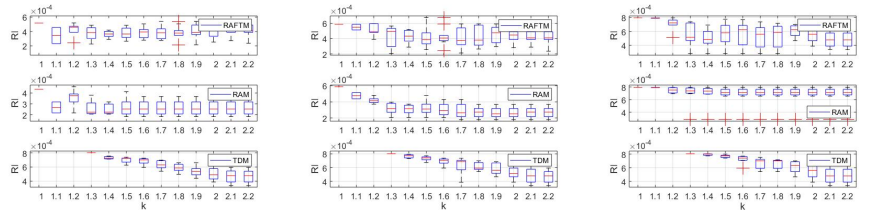
Fig. 3: Energy Consumption (EC) gain: $\overline{\mathcal{N}} = 10$, $M = 4$ (a, b, c), $\overline{\mathcal{N}} = 20$, $M = 4$ (d, e, f), and $\overline{\mathcal{N}} = 20$, $M = 6$ (g, h, i).

and $k = 1.6$ ($\overline{\mathcal{N}} = 20$ and $M = 6$), RAFTM has a similar behaviour with TDM, since it decides to duplicate the majority of the tasks, achieving similar gains, as explained in the next section.

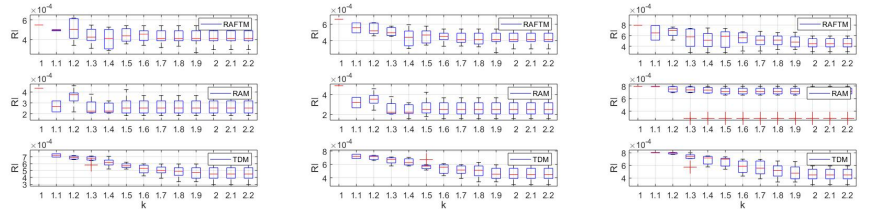
Regarding the comparison with RAM, in this experimental set-up, RAFTM and RAM have the same feasibility, since the reliability constraints can always be met by executing only the original task with a high processor frequency. Note that, we explore the difference of feasibility between RAFTM and RAM, in the experimental set-up 2, where high frequencies cannot always satisfy reliability thresholds. We observe that the feasibility of RAFTM and RAM is not modified as the number of cores is increased, due to the dependencies of the task graph. However, since TDM requires to duplicate all tasks, its feasibility is increased with more processors, approaching the RAFMT and RAM feasibility with 6 processors.



(a) TL-DVFS ($\overline{\mathcal{N}} = 10$, $M = 4$) (b) PL-DVFS ($\overline{\mathcal{N}} = 10$, $M = 4$) (c) SL-DVFS ($\overline{\mathcal{N}} = 10$, $M = 4$)



(d) TL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 4$) (e) PL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 4$) (f) SL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 4$)



(g) TL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 6$) (h) PL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 6$) (i) SL-DVFS ($\overline{\mathcal{N}} = 20$, $M = 6$)

Fig. 4: Reliability improvement: $\overline{\mathcal{N}} = 10$, $M = 4$ (a,b,c), $\overline{\mathcal{N}} = 20$, $M = 4$ (d,e,f), and $\overline{\mathcal{N}} = 20$, $M = 6$ (g,h,i).

4.2.2 Energy consumption

The energy consumption of RAM and TDM, compared to the proposed approach, is depicted in Fig. 3. The minimum, average and maximum gains

Table 4: Min, avg. and max energy gains (%)

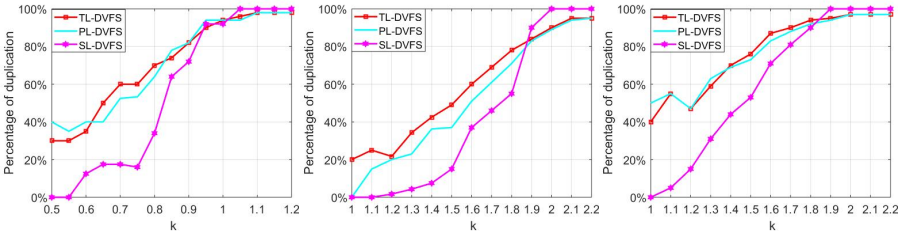
		TL-DVFS			**PL-DVFS**			**SL-DVFS**		
$\overline{\mathcal{N}}$	**M**	**Min**	**Avg.**	**Max**	**Min**	**Avg.**	**Max**	**Min**	**Avg.**	**Max**
RAM - RAFTM										
10	4	4.8	36.65	55.1	2.11	31.46	52.64	0	42.90	78.31
20	4	1.16	29.19	57.35	0	23.25	54.87	0	42.2	105.33
20	6	9.76	38.97	59.14	8.63	36.72	58.84	0	64.29	105.94
TDM - RAFTM										
10	4	0	47.34	128.52	0	56.52	177.58	0	65.86	170.68
20	4	0	49.67	123.05	0	67.3	148.21	0	63.94	132.08
20	6	0	30.54	91.21	0	37.15	116.08	0	53.09	115.16

are depicted in Table 4. Note that, the minimum gain is 0 between TDM and RAFTM, when the deadlines are less strict. When deadline is relaxed, RAFTM performs duplication for all tasks, as TDM. For RAM, we observe a minimum gain of 0, only in very few strict deadlines, either with a high number of tasks or due to SL-DVFS, where the frequency assignment is very restricted. In the strict deadlines, RAFTM and RAM have a similar behavior: applying a high frequency to meet the timing constraint, thus no duplication is possible. From the obtained results, we can make the following general observations:

- When the number of tasks is increased from $\overline{\mathcal{N}} = 10$ to $\overline{\mathcal{N}} = 20$ with the same number of cores (from Fig. 3a to Fig. 3d, from Fig. 3b to Fig. 3e, and from Fig. 3c to Fig. 3f), the energy savings remain high for the proposed RAFTM approach, compared to RAM and TDM. We observe a slight decrease in average savings regarding RAM in TL-DVFS and PL-DVFS schemes, due to the fact that RAFTM behaves as RAM in very strict deadlines. On the one hand, RAM consumes more energy compared to RAFTM, as Table 4 shows. On the other hand, RAFTM is able to find solutions in more cases, compared to TDM, especially for strict deadlines, as Fig. 1 shows.
- When the number of cores increases from $M = 4$ to $M = 6$, with the same number of tasks $\overline{\mathcal{N}} = 20$, (from Fig. 3d to Fig. 3g, from Fig. 3e to Fig. 3h, and from Fig. 3f to Fig. 3i), the energy savings of RAFTM compared to RAM are enlarged, on average, to 38.9% (TL-DVFS), 36.72% (PL-DVFS), and 64.29% (SL-DVFS). This is supported by the fact that when more resources like processors are available, our proposed approach has more freedom to use the available processors when performing the task mapping, minimizing the total energy consumption.
- Among different DVFS schemes, we observed that SL-DVFS has a higher impact on the observed gains of RAFTM vs RAM, compared to the impact it has on the observed gains of RAFTM vs TDM, as the number of tasks and cores increases. When the supported DVFS scheme is flexible, RAM performs a more fine-grained assignment, achieving a lower energy consumption. However, when SL-DVFS is supported, RAM is obliged to select a high frequency in order to meet the highest reliability threshold of the tasks and executes all tasks with this high frequency, causing large energy consumption. On the other hand, the proposed RAFTM can exploit task duplication, applying a lower frequency, and thus, reducing the energy consumption, when time slack is available for relaxed deadline cases.

With a more detailed observation, when both RAFTM and TDM approaches can find solutions, RAFTM provides a solution that consumes significantly less energy. More precisely, when $\overline{\mathcal{N}} = 20$ and $M = 4$, for TL-DVFS we observe a maximum gain of 123.05%, with an average gain of 49.67%, for PL-DVFS, we observe a maximum gain of 148.21% with an average gain of 67.3%, and, for SL-DVFS, there is a maximum gain 132.08%, with an average gain of 63.94%.

Furthermore, RAFTM finds a solution, that consumes less energy than RAM, since RAFTM can duplicate some tasks in order to use lower frequencies, reducing energy consumption. When the deadline is not so strict (part of graph with almost flat average gain in the box-plot of Fig. 3), the highest energy gains are observed. More precisely, when $M = 4$, we observe an average gain of 54.01% (TL-DVFS), 50.5% (PL-DVFS), and 73.48% (SL-DVFS) for $\overline{\mathcal{N}} = 10$ and an average gain of 54.27% (TL-DVFS), 48.88% (PL-DVFS), and 92.19% (SL-DVFS) for $\overline{\mathcal{N}} = 20$. When $M = 6$ and $\overline{\mathcal{N}} = 20$, the observed average gain is 57.98% (TL-DVFS), 57.06% (PL-DVFS), and 102.11% (SL-DVFS). For very strict deadlines, the energy gains of RAFTM and RAM are similar; RAFTM behaves as RAM, since there is no available time slack, and thus, RAFTM assigns high frequencies, without task duplication. We observe minimum gains when $k = 0.5$ ($\overline{\mathcal{N}} = 10$) and $k = 1$ ($\overline{\mathcal{N}} = 20$). When SL-DVFS and less cores are used ($M = 4$), RAFTM cannot exploit duplication and use different frequencies to reduce energy consumption, and thus, it behaves as RAM. When time slack exists, RAFTM can take advantage of it and duplicate some tasks, if energy gains can be achieved.



(a) $\overline{\mathcal{N}} = 10$, $M = 4$ (b) $\overline{\mathcal{N}} = 20$, $M = 4$ (c) $\overline{\mathcal{N}} = 20$, $M = 6$

Fig. 5: RAFTM duplication percentage: a) $\overline{\mathcal{N}} = 10$, $M = 4$, b) $\overline{\mathcal{N}} = 20$, $M = 4$, and c) $\overline{\mathcal{N}} = 20$, $M = 6$ (all DVFS schemes).

4.2.3 Reliability Improvement

We also compare the reliability improvement with regard to the reliability threshold in Fig. 4, for the three DVFS schemes and for $\overline{\mathcal{N}} = 10$ and $\overline{\mathcal{N}} = 20$, with M=4, and $\overline{\mathcal{N}} = 20$, with M=6. Generally:

– Regarding RAM, when TL-DVFS is considered, it has the lowest reliability improvement. This is because RAM only requires to meet the reliability threshold in order to find a solution. However, when PL-DVFS and SL-DVFS are considered, RAM is obliged to select a higher frequency, even for tasks with lower reliability threshold, due to the restrictions in the DVFS schemes. As a result, the reliability improvement of RAM is increased, when the DVFS schemes are more restricted. Note that, when the number of tasks

is increased in SL-DVFS, RAM is obliged to always select high frequencies to meet the highest reliability threshold, among all tasks, and then, use this frequency for task execution, even if time slack exists.
– Regarding RAFTM, it provides higher reliability improvements than RAM, for all deadlines. For tight deadlines, it provides lower reliability improvements than TDM, as it partially duplicates the task set. However, as discussed in the previous section, RAFTM is more capable of finding solutions compared to TDM, and with significantly reduced energy consumption. When the deadline is not so strict, it provides the same reliability improvements as TDM, since they behave in a similar way.
– Regarding TDM, since it always duplicates the tasks, it provides a high reliability improvement, if it can find a solution. Therefore, going from TL-DVFS to PL-DVFS and SL-DVFS, does not have a high impact on the reliability improvement. Moreover, when a solution is found in strict deadlines, it has typically significant reliability improvement, at the price of high energy consumption, since high frequencies are required to meet the strict timing constraints.

Table 5: Running time (seconds), when at least one solution is found, with $\overline{\mathcal{N}} = 10$, $M = 4$

	deadline(k)	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1	1.05	1.1	1.15	1.2
TL DVFS	RAFTM	0.45	0.57	0.95	2.17	3.31	4.31	3.77	3.11	5.57	1.47	1.92	3.71	0.51	0.3	0.27
	RAM	0.3	0.25	0.27	0.22	0.43	0.33	0.16	0.16	0.21	0.18	0.18	0.16	0.19	0.16	0.19
	TDM	2.52	0.56	0.61	0.97	1.53	2.65	2.45	2.08	1.26	1.08	0.72	0.72	0.55	0.36	0.37
PL DVFS	RAFTM	1.65	6.03	8.2	15.26	24.06	32.3	34.38	40.76	29.12	29.05	13.9	13.54	10.94	4.19	3.51
	RAM	0.82	1.78	1.51	1.7	1.94	1.72	1.79	1.79	2.15	2.22	1.87	1.88	2.1	1.77	1.84
	TDM	2.55	2.27	1.7	4.45	5.97	9.17	5.04	4.97	4.83	3.61	3.22	2.47	2.34	1.84	1.87
SL DVFS	RAFTM	0.28	0.19	0.26	0.83	1.47	1.86	1.18	1.13	0.84	0.92	0.67	0.59	0.53	0.27	0.23
	RAM	0.11	0.08	0.11	0.09	0.19	0.2	0.2	0.07	0.09	0.08	0.08	0.09	0.09	0.08	0.08
	TDM	0.48	0.14	1.02	1.64	1.51	2.5	1.83	1.59	1.43	1.7	0.88	0.68	0.71	0.54	0.44

Table 6: Running time (seconds), when at least one solution is found, with $\overline{\mathcal{N}} = 20$, $M = 4$

	deadline(k)	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0	2.1	2.2
TL DVFS	RAFTM	14.9	61	74.9	138.2	202.1	256.9	327.3	219	1039.3	1054.7	126.8	58.7	123.1
	RAM	0.07	0.04	0.067	0.054	0.041	0.061	0.066	0.045	0.043	0.036	0.053	0.052	0.044
	TDM	-	-	-	9188.3	1550.4	1015.9	1165.4	2304.7	2814.6	1350.6	700.6	549.3	237.8
PL DVFS	RAFTM	228.6	662.2	529.9	1092	1815	4665	15722	4676	33294	70465	24239	24054	24072
	RAM	84.3	99.2	44.4	74.3	76.5	65.5	52.4	55	56.3	41.7	42.8	57.4	56.3
	TDM	-	-	-	2740	5677.5	5084.1	9641.1	14770.4	10571.4	8235.2	5517.1	3619.0	3169.6
SL DVFS	RAFTM	0.53	0.3	5.76	14.14	21.69	98.22	91.88	183.39	1301.28	585.03	65.29	28.56	15.08
	RAM	0.08	0.06	0.038	0.069	0.045	0.066	0.059	0.036	0.027	0.026	0.025	0.037	0.025
	TDM	-	-	-	238.51	230.59	230.59	230.59	230.59	230.59	230.59	230.59	230.59	230.59

Table 7: Running time (seconds), when at least one solution is found, with $\overline{\mathcal{N}} = 20$, $M = 6$

	deadline(k)	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0	2.1	2.2
TL DVFS	RAFTM	0.8	58.1	26.9	89.9	81.9	109.8	58.6	62.8	59.8	14.7	2.5	0.7	0.6
	RAM	0.07	0.06	0.08	0.09	0.10	0.06	0.05	0.05	0.07	0.05	0.06	0.08	0.05
	TDM	-	70.8	333.4	124.4	298.2	227.4	70.5	160.6	203.4	39.8	2.9	1.4	0.9
PL DVFS	RAFTM	114.2	697.3	730.8	661.5	859.5	824.0	1090.1	503.3	569.2	188.6	95.9	82.5	75.0
	RAM	566.0	397.9	533.6	671.5	546.1	820.5	954.5	562.0	687.4	693.2	799.0	1113.8	700.0
	TDM	-	2861.2	2874.1	5312.8	8795.7	1477.3	1237.6	1243.7	1794.6	232.5	41.7	42.2	35.2
SL DVFS	RAFTM	0.4	0.3	1.4	12.1	12.6	23.5	19.2	19.7	16.1	21.3	5.9	4.7	5.6
	RAM	0.09	0.07	0.06	0.12	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.05	0.04
	TDM	-	8.7	111.0	37.8	71.9	83.0	52.3	52.6	41.9	32.8	19.5	20.2	16.3

4.2.4 RAFTM duplication

Fig. 5 depicts the task duplication obtained by the proposed approach. RAM approach does not apply duplication for any task (0%) and TDM duplicates all tasks (100%). Generally observed for all DVFS schemes, when the deadline is more relaxed, RAFTM can duplicate more tasks using lower frequency, thus achieving less energy consumption, due to the available time slack. Furthermore, as all processors have the same frequency in SL-DVFS, this DVFS scheme has less flexibility in task duplication and scheduling, when deadlines are strict. In Fig. 5, there are two interesting observations: i) for some points, the duplication percentage decreases, when deadline is relaxed (Fig. 5b, when $k = 1.2$ for TL-DVFS, and Fig. 5c, when $k = 1.2$ for both TL/PL-DVFS), and ii) the duplication percentage does not reach 100% for TL/PL-DVFS, as SL-DVFS does. The reason is the flexibility to decide the task frequency for different DVFS schemes, and the reliability threshold's value. When the task's reliability threshold can be satisfied without duplication and with a low frequency, and energy consumption is less than the energy consumption when duplicating with the lowest frequency, then TL-DVFS and PL-DVFS schemes do not duplicate the task. However, with SL-DVFS, the same frequency must be assigned to all processors. Hence, a task with a high reliability constraint forces other tasks with low reliability constraints to execute with a higher frequency than the required one. As a result, duplication does not provide benefits, even if time slack exists. Due to these reasons, the duplication percentage is usually smaller for SL-DVFS, compared to TL/PL-DVFS, except for relaxed enough deadlines.

4.2.5 Time

Tables 5 to 7 provide the time required to find a solution, when $\overline{\mathcal{N}} = 10$ and $\overline{\mathcal{N}} = 20$ with $M = 4$, and $\overline{\mathcal{N}} = 20$ with $M = 6$. Comparing Table 6

and Table 5, with more tasks, more time is required to find a solution, as expected. For RAFTM, when deadline is very strict or very relaxed, less time is needed to obtain the solutions. However, with intermediate deadlines, more time is required as RAFTM needs to explore the available time slack to decide which, and how many, tasks to be duplicated, without violating constraints, while consuming the least energy. TDM is the most running time expensive approach, because all tasks need to be duplicated, increasing the number of tasks to be scheduled, and thus, the time to find the solutions. Although RAM is the least running time expensive approach, it provides less energy savings.

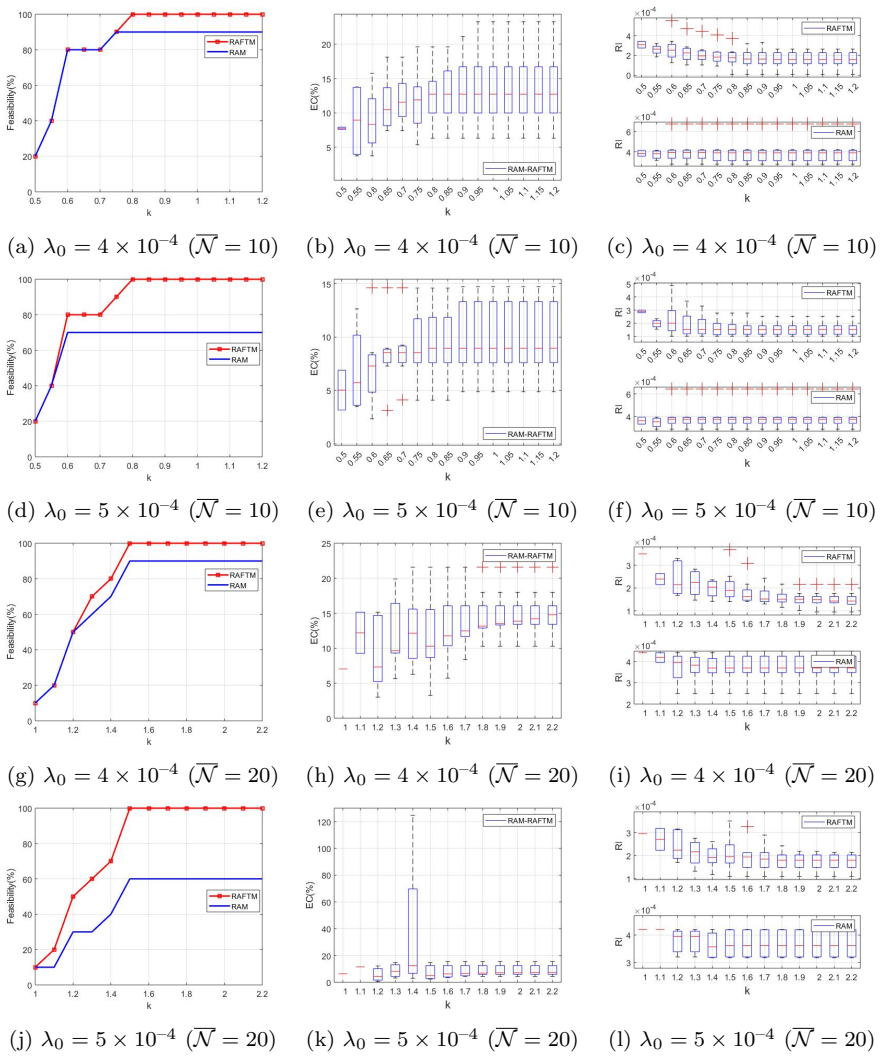
4.3 Experimental set-up 2

In this experimental set-up, we explore the behavior of RAFTM and RAM at different failure rates λ_0 under TL-DVFS scheme. First, we extend the range of the task reliability threshold, in order to have few tasks with higher reliability requirements. Then, we tune the parameter λ_0 in order to change the failure rate. This experimental set-up shows how the fault model influences the ability of obtaining feasible solutions, through a slight increase of λ_0. The results show that the proposed approach provides better results than RAM, with the fault rate changing. These modifications will affect only the proposed RAFTM and RAM approaches, since they use the reliability constraint to decide the task mapping. The results of TDM remain similar to the previous section.

4.3.1 Feasibility

The first column of Fig. 6 depicts the ability of RAM, compared to RAFTM, to obtain feasible solution, for $\lambda_0 = 4{\times}10^{-4}$ (Fig. 6a and Fig. 6c) and $\lambda_0 = 5{\times}10^{-4}$ (Fig. 6b and Fig. 6d), with $\overline{\mathcal{N}} = 10$ and $\overline{\mathcal{N}} = 20$, $M = 4$. Generally:

- As the λ_0 increases, the reliability of a task, due to the processor fault rate, is decreased. Therefore, with λ_0 increasing, the capability of RAM to always find a frequency, that meets the reliability threshold of all tasks, decreases, especially for tasks with high reliability requirements. Furthermore, even with λ_0 increasing, RAFTM is able to still meet these high reliability requirements, by duplicating tasks and assigning a high frequency. This can be observed by comparing Fig. 6a to Fig. 6d, where the feasibility of RAM is reduced by 20%, and Fig. 6g to Fig. 6j, where the feasibility of RAM is reduced by 30%. However, RAFTM feasibility remains the same, always finding a solution at relaxed deadlines.
- As the number of tasks increases, the feasibility is affected. As we observe by comparing Fig. 6a to Fig. 6g and Fig. 6d to Fig. 6j, the curves of RAM and RAFTM separate at stricter deadlines, when $\overline{\mathcal{N}} = 20$, compared to the ones when $\overline{\mathcal{N}} = 10$. Comparing the feasibility between set-up 1 and set-up 2, we observe that the feasibility of set-up1 can be 100% for both RAFTM and RAM, with a smaller average fault rate λ_0 at relaxed deadlines. However, with the average fault rate increasing, it is not possible for RAM to



(a) $\lambda_0 = 4 \times 10^{-4}$ ($\overline{\mathcal{N}} = 10$) (b) $\lambda_0 = 4 \times 10^{-4}$ ($\overline{\mathcal{N}} = 10$) (c) $\lambda_0 = 4 \times 10^{-4}$ ($\overline{\mathcal{N}} = 10$)

(d) $\lambda_0 = 5 \times 10^{-4}$ ($\overline{\mathcal{N}} = 10$) (e) $\lambda_0 = 5 \times 10^{-4}$ ($\overline{\mathcal{N}} = 10$) (f) $\lambda_0 = 5 \times 10^{-4}$ ($\overline{\mathcal{N}} = 10$)

(g) $\lambda_0 = 4 \times 10^{-4}$ ($\overline{\mathcal{N}} = 20$) (h) $\lambda_0 = 4 \times 10^{-4}$ ($\overline{\mathcal{N}} = 20$) (i) $\lambda_0 = 4 \times 10^{-4}$ ($\overline{\mathcal{N}} = 20$)

(j) $\lambda_0 = 5 \times 10^{-4}$ ($\overline{\mathcal{N}} = 20$) (k) $\lambda_0 = 5 \times 10^{-4}$ ($\overline{\mathcal{N}} = 20$) (l) $\lambda_0 = 5 \times 10^{-4}$ ($\overline{\mathcal{N}} = 20$)

Fig. 6: TL-DVFS: Feasibility (first column), Energy Consumption (EC) gain (second column) and Reliability Improvement (RI) (third column) for $\lambda_0 = 4 \times 10^{-4}$ and $\lambda_0 = 5 \times 10^{-4}$, with $\overline{\mathcal{N}} = 10$ and $\overline{\mathcal{N}} = 20$ ($M = 4$).

find a solution satisfying the reliability requirements, even with maximum frequency and relaxed deadline. This observation worsens with λ_0 increasing. Thus, with a different fault model, executing only the original task may not be able to provide reliable execution. Therefore, replication is needed in order to guarantee the reliability threshold of the tasks.

4.3.2 Energy consumption

The second column of Fig. 6 depicts the energy consumption of RAM, compared to the proposed approach, when RAM found a solution.

- As the λ_0 increases, the processor sensibility to faults is increased. Then, the energy gains are slightly reduced, from 12.05% to 8.87% ($\overline{\mathcal{N}} = 10$, Fig. 6b to Fig. 6e) and from 12.60% to 10.75% ($\overline{\mathcal{N}} = 20$), Fig. 6h to Fig. 6k, since RAFTM also selects a high frequency, in order to meet the high reliability thresholds of tasks.
- As the number of tasks increases, the gains in energy consumption remain similar. As we observe by comparing Fig. 6b to Fig. 6h and Fig. 6e to Fig. 6k, the energy savings are decreased, compared to set-up 1. For instance, from 36.65% (set-up 1) to 12.05% and 8.87%, when $\overline{\mathcal{N}} = 10$ and $M = 4$. This is due to the fact that, based on $\lambda(f)$ (see Section 2.3), the reliability with same frequency decreases, when λ_0 increases. In order to guarantee a reliable execution, a high frequency must be assigned to meet the reliability requirement, leading to higher energy consumption for the proposed approach.

4.3.3 Reliability improvement

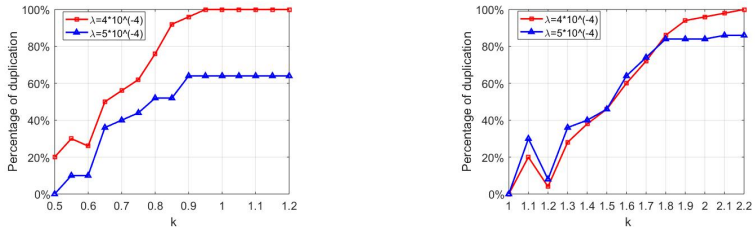
The third column of Fig. 6 depicts the reliability improvement of RAFTM and RAM, for the cases where RAM was able to find a solution. On the basis of satisfying the reliability requirements, RAM achieves a slightly better reliability improvement than our approach, while it sacrifices the ability of obtaining feasible solutions.

Table 8: Running time (seconds), when at least one solution is found, with $\overline{\mathcal{N}} = 10$, $M = 4$ (TL-DVFS)

	deadline(k)	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1	1.05	1.1	1.15	1.2
$\lambda_0 = 4 \times 10^{-4}$	RAFTM	0.77	0.93	1.01	1.49	1.48	2.00	1.28	1.31	1.21	0.74	0.68	0.66	0.62	0.64	0.62
	RAM	0.07	0.04	0.06	0.07	0.13	0.19	0.12	0.05	0.06	0.05	0.06	0.06	0.05	0.04	0.04
$\lambda_0 = 5 \times 10^{-4}$	RAFTM	0.67	0.61	0.87	1.18	0.87	1.39	0.80	0.72	0.66	0.55	0.53	0.58	0.55	0.57	0.50
	RAM	0.10	0.05	0.05	0.06	0.13	0.20	0.13	0.05	0.05	0.08	0.05	0.05	0.05	0.04	0.07

Table 9: Running time (seconds), when at least one solution is found, with $\overline{\mathcal{N}} = 20$, $M = 4$ (TL-DVFS)

	deadline(k)	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0	2.1	2.2
$\lambda_0 = 4 \times 10^{-4}$	RAFTM	26.85	194.76	98.45	208.45	224.04	315.23	167.25	326.79	278.74	99.46	216.81	31.26	4.79
	RAM	0.04	0.03	0.02	0.05	0.03	0.04	0.04	0.04	0.03	0.02	0.02	0.03	0.02
$\lambda_0 = 5 \times 10^{-4}$	RAFTM	32.08	206.57	583.86	156.99	155.10	117.55	155.63	174.00	19.15	24.26	33.65	3.52	3.06
	RAM	0.05	0.03	0.03	0.05	0.04	0.05	0.05	0.04	0.04	0.02	0.03	0.04	0.02



(a) $\overline{\mathcal{N}} = 10$ (b) $\overline{\mathcal{N}} = 20$

Fig. 7: RAFTM duplication percentage (TL-DVFS): a) $\overline{\mathcal{N}} = 10$, and b) $\overline{\mathcal{N}} = 20$, for $\lambda_0 = 4 \times 10^{-4}$ and $\lambda_0 = 5 \times 10^{-4}$, with $M = 4$.

4.3.4 RAFTM Duplication

The percentage of duplicated task is depicted in Fig. 7. As Based on the fault model in Section 2.3, the reliability decreases with higher λ_0. When the timing constraints are always satisfied, a higher λ_0 with the same frequency, leads to lower reliability. To meet reliability constraints, either a very high frequency is assigned to original tasks, or a relative high frequency is required for both original and duplicated tasks. The first option consumes less energy. As observed in Fig. 7a and Fig. 7b, with relaxed deadlines, RAFTM duplicates more tasks when $\lambda_0 = 4 \times 10^{-4}$.

4.3.5 Time

Table 8 and 9 give the time needed to obtain a solution for our proposed RAFTM and RAM after the extension of fault rate λ_0. Similar to the time analysis in set-up 1, RAM needs less running time than our proposed approach when finding a feasible solution, at the price of more energy consumption and less ability to obtain feasible solutions.

5 Conclusion

This work proposes a RAFTM approach for dependent tasks and homogeneous platforms, under several DVFS schemes. It jointly decides the frequency assignment to tasks, task allocation to processors and tasks to be duplicated, minimizing energy consumption, under real-time and reliability constraints. RAFTM approach is compared with a reliability-aware task mapping approach and a fault-tolerant task mapping approach, under several scenarios. The results show that RAFTM finds solutions, when other approaches fail. TL-DVFS achieves less energy consumption, due to flexibility in task frequency assignment, compared to SL-DVFS. RAFTM provides more energy savings, and, at the same time, higher feasibility, without violating timing and reliability constraints, under the possible DVFS schemes. In future, we will extend RAFTM for heterogeneous platforms and controlled heuristics.

References

1. G. Xie, Y. Chen, X. Xiao *et al.*, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 3, pp. 167–181, 2018.
2. B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 316–328, 2010.
3. M. A. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 813–825, 2017.
4. B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," *ACM Transactions on Design Automation of Electronic Systems*, vol. 18, no. 2, pp. 1–21, 2013.
5. L. Zhang, K. Li, Y. Xu *et al.*, "Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster," *Information Sciences*, vol. 319, 2015.
6. J. Zhou, J. Sun, X. Zhou *et al.*, "Resource management for improving soft-error and lifetime reliability of real-time MPSoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2215–2228, 2019.
7. S. Wang, K. Li, J. Mei *et al.*, "A reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems," *Journal of Grid Computing*, vol. 15, no. 1, p. 23–39, Mar. 2017.
8. G. Xie, Y. Chen, Y. Liu *et al.*, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1629–1640, 2017.
9. L. Zhang, K. Li, K. Li *et al.*, "Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems," *International Journal of Electrical Power & Energy Systems*, vol. 78, pp. 499–512, 2016.
10. Y. Guo, D. Zhu, and H. Aydin, "Reliability-aware power management for parallel real-time applications with precedence constraints," in *IEEE International Green Computing Conference and Workshops*, 2011, pp. 1–8.
11. K. Huang, X. Jiang, X. Zhang *et al.*, "Energy-efficient fault-tolerant mapping and scheduling on heterogeneous multiprocessor real-time systems," *IEEE Access*, vol. 6, pp. 57 614–57 630, 2018.
12. M. Salehi, M. K. Tavana, S. Rehman *et al.*, "DRVS: Power-efficient reliability management through dynamic redundancy and voltage scaling under variations," in *IEEE/ACM ISLPED*, 2015, pp. 225–230.
13. M. Salehi, A. Ejlali, and B. M. Al-Hashimi, "Two-phase low-energy n-modular redundancy for hard real-time multi-core systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1497–1510, 2016.
14. S. Tosun, "Energy- and reliability-aware task scheduling onto heterogeneous MPSoC architectures," *The Journal of Supercomputing*, vol. 62, 2012.
15. C. Gou, A. Benoit, M. Chen *et al.*, "Reliability-aware energy optimization for throughput-constrained applications on MPSoC," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, 2018, pp. 1–10.
16. G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 3, 2014.
17. Z. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," *IEEE/ACM International Conference on Computer Aided Design*, pp. 35–40, 2004.
18. K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, p. 94–125, Mar. 2004.
19. S. Rokicki, D. Pala, J. Paturel *et al.*, "What you simulate is what you synthesize: Designing a processor core from c++ specifications," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019.

20. G. Quan and V. Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 329–339, 2010.
21. M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Int. Workshop on Workload Characterization*, 01 2002, pp. 3 – 14.
22. J. Deverge and I. Puaut, "Safe measurement-based wcet estimation," in *WCET*, 2007.