Energy-Efficient Partial-Duplication Task Mapping under multiple DVFS schemes

Minyu Cui, Angeliki Kritikakou, Lei Mo, Emmanuel Casseau

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

Energy-Efficient Partial-Duplication Task Mapping 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 savings, but with a negative impact on reliability, especially when the applied frequency is low. Using high frequencies, required to meet reliability constraints, or replicating tasks increases energy consumption. To reduce energy consumption, while enhancing reliability and satisfying real-time constraints, we propose a hybrid approach that combines distinct reliability enhancement techniques, under task-level, processor-level and system-level DVFS. Our task mapping problem jointly decides task allocation, task frequency assignment, and task duplication, under real-time and reliability constraints. This is achieved by formulating the task mapping problem as a Mixed Integer Non-Linear Programming (MINLP) problem, and equivalently transforming it into a Mixed Integer Linear Programming (MILP), that can be optimally solved. From the obtained results, the proposed approach achieves better energy consumption, finding solutions, when replication approaches fail.

Keywords Fault tolerant · Task mapping · DVFS · Real-time execution · Reliability · Energy minimisation

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

The execution of parallel applications on multicore platforms requires both a timely and a reliable execution of tasks. In time-critical domains, tasks must be finished before a given deadline, in order to consider the result as correct. Meanwhile, the correct execution of tasks can be threatened by several sources, such as radiation [1] and electromagnetic interference [2], causing non-persistent *transient faults* (also called *soft errors*). Due to the technology size reduction and the increasing scaling of CMOS technology, systems have become more susceptible to soft errors [3]. To enable correct execution of the tasks, the system reliability must be enhanced. To this end, typical methods include the use of high frequencies to execute the tasks [4], and system-level fault-tolerance techniques, where tasks are replicated [5–7]. Nonetheless, these approaches both increase the system energy consumption. Energy saving has become one of the essential objectives during system deployment, due to its impact on the environment and lifetime of the system. In order to enhance energy saving, Dynamic Voltage/Frequency Scaling (DVFS) has been adopted in multicore platforms as an efficient way to adjust the voltage and frequency for task 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 [5]. The system becomes more susceptible to soft errors when the frequency is low [8]. Therefore, adequate *mapping approaches* are required in order to obtain a reliable, in-time and energy efficient task execution on multicore platforms.

Although both reliability and energy management have been extensively (but often independently) studied, their co-management, during system deployment, has been addressed only recently [9]. One category of such approaches focus on meeting the reliability requirements, considering only original tasks [10, 11]. However, these methods usually assign high frequencies, whereas it may be impossible to satisfy the reliability constraints, even with the highest platform frequency. The other category replicates tasks to improve reliability. Full replication, such as [7], leads to large energy consumption, combined with a negative impact on execution time; the end-times of tasks are delayed, due to the execution of task replicas. Partial task replication is usually performed with heuristics, as in [6,12], without considering timing constraints.

The main contribution of this work is to propose an optimal task mapping approach, which keeps the benefits and withdraws the drawbacks of existing approaches. The goal is to minimize energy consumption, by deciding between reliable original task execution and task duplication, under real-time and reliability constraints, exploiting different DVFS schemes. 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 tasks that require duplication is reduced, relieving the negative impact on execution time and energy consumption. The proposed approach optimally and concurrently decides which tasks to be duplicated, the frequency per task and the task allocation, under both real-time and reliability constraints. Unlike the majority of State-of-the-

Art (SoA) techniques, original and duplicated tasks can have different operating frequencies, being more suitable for real-world systems [13]. To achieve that, the task mapping problem is formulated as an Mixed Integer Non-Linear Programming (MINLP), and safely transformed into an Mixed Integer Linear Programming (MILP) problem, in order to be solved optimally. This work extends [14], where only frequency assignment per task (TL-DVFS) has been considered and a less efficient solver has been used. The proposed energy-aware fault tolerant task mapping approach leverages frequency assignment per processor (PL-DVFS) and frequency assignment per system (SL-DVFS). Extensive experiments have been performed by using Gurobi solver which has a better ability for solving optimization problems, under different scenarios. Experiments show the impact of the flexibility of the three DVFS schemes on feasibility, energy savings, reliability achievement and time required to find solutions. Overall, from the obtained results, the proposed approach can find feasible solutions, when existing replication approaches sometimes fail, and provides solutions consuming less energy, compared to existing approaches.

The rest of the paper is organized as follows: Section 2 presents the relevant related works and Section 3 describes a motivational example. Section 4 introduces the system model. Section 5 presents the problem formulation and the proposed duplication-based Energy-efficient Fault-tolerant Task Mapping Approach (EFTMA). Section 6 presents the evaluation results. Finally, Section 7 concludes this work.

2 Related Work

Table 1 summarises existing task mapping approaches with DVFS technique, under Energy Budget (EB), Real-Time (RT) and Reliability (R) constraints, with the goal to maximizing system Reliability (mR), minimizing schedule length (also called makespan) (mS) of the task scheduling or minimizing Energy consumption (mE). 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 [7, 15]. The platform has Homogeneous (HO), Heterogeneous (HE) or Single (S) processor(s). Based on the problem formulation and solving method, solutions are Feasible (F) or Optimal (O).

Regarding reliability maximization, approaches without fault tolerance map only original tasks, e.g., through online heuristics [16]. Approaches with fault-tolerance i) insert a recovery task, always executed with maximum frequency if an error occurs during original task execution [5], ii) map a given number of task replicas [13], and iii) decide the number of replicas [6].

Regarding minimizing schedule length, the optimal scheduling is obtained using Integer Linear Programming (ILP) on a distributed platform composed of heterogeneous processing units [17]. In [18], an algorithm based on a weighted-based mechanism pre-assigns energy consumption to unassigned tasks with the goal of minimizing the schedule length for energy consumption constrained parallel applications, and the idea is extended to solve reliability maximiza-

tion problems. However, no fault tolerance techniques are considered in these above works. In [19], the authors studied the problem of makespan optimization jointly considering reliability, temperature and stochastic characteristics of precedence-constrained tasks. An affinity-driven task allocation and scheduling approach is proposed and a heuristic is designed by assigning a task to the processor with highest affinity.

Regarding minimizing energy consumption, which is the goal in this work, approaches exist without fault tolerance, e.g. by scheduling independent tasks on heterogeneous platform using DVFS [28] or by decomposing the task mapping problem into two sub-problems (satisfying reliability goal and minimizing resource consumption) [10, 20] and by applying heuristics [11]. 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, which leads to some time slack. Using DVFS technique, the available time slack is reserved to execute a recovery task (individual [21] or shared [9]) with the maximum frequency, to preserve reliability. If an error is detected, the recovery task is evoked. A longest-task-first heuristic applies DVFS and recovery tasks only to a selected task subset [22]. However, such approaches usually use high frequencies to execute the original tasks to meet reliability requirements, 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, aiming at energy savings, apply replication schemes to provide fault-tolerance. Some techniques compute the required number of replicas to always meet reliability constraints [7, 12]. Heuristics are typically proposed, due to high computation complexity or NP-hard problems, e.g., a

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

Ref.	Goal			Fault tol.		Platform			Constraints			Sol.	
	mE	mR	mS	Rec.	Rep.	HO	HE	S	RT	EB	R	F	O
[16]		√					√			√		√	
[13]		√			√	√			√	√		√	
[5]		√		√				√	√	√			√
[6]		√			√		√				√	√	
[17]			√				√		√				√
[18]		(√)	√				√			√		√	
[19]			√		√		√		√		√	√	
[10], [20]	√						√				√	√	
[11]	√						√		(√)		√	√	
[21], [22]	√			√			√		√		√	√	
[9]	√			√				√	√		√	√	
[23]	√			(√)			√		√		√	√	
[7], [15]	√				√	√			√		√	√	
[12]	√				√		√				√	√	
[24]	√				√	√			(√)		(√)	√	
[25]	√				√	√			√		√	√	
[26]	√			√	√	√			√			√	
[27]	√				√		√				(√)		√
[14]	√				√	√			√		√		√
[28]	√						√		√			√	
Prop.	√				√	√			√		√		√

first-fit decreasing heuristic for homogeneous platforms [7] and a list scheduling heuristic for heterogeneous platforms [12]. The work in [15] improves the approach proposed in [7] by applying a layered Worst-Fit-Decreasing strategy, where the secondary replicas are executed with same frequency. Hybrid approaches use a given number of replicas to explore the available time slack to impose energy overhead, when an error occurs [25]. 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 [24], a heuristic decides among single execution, task duplication and task triplication. However, no guarantees are provided on real-time and reliability constraints. In [26], 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 [27] maps independent tasks on a heterogeneous platform to satisfy a given percentage of duplicated tasks, under a energy budget constraint. However, reliability is not considered.

This work proposes an optimal task mapping energy minimisation approach for homogeneous multicore platforms with DVFS capabilities, that combines partial task duplication and reliable original task execution, under real-time and reliability constraints. Furthermore, this work exploits the impact of hardware DVFS schemes over such task mapping methods.

3 Motivational example

Initially, we show the benefits of the proposed Energy-efficient Fault-tolerant Task Mapping Approach (EFTMA) through a motivational example. We consider a simple case with a single task, under the TL-DVFS scheme, with C=4×10^8, a reliability threshold equals to 0.9995 and the first five voltage/frequency levels of the platform used in the experimental section (Table 4). Table 2 depicts the possible solutions (Sol.), with the selected frequency (f), the execution time (t), energy consumption ($E = P_l \times t$) and reliability (R). Columns f_1-f_5 corresponding to the execution of only the original task, while columns f_1/f_1-f_5/f_5 corresponding to the execution of both the original and its replica. Table 2.a enumerates all feasible solutions that satisfy only the reliability constraint, in the execution time increasing order, for i) the proposed approach (EFTMA), ii) the Reliability-Aware Mapping (RAM) approach, which satisfies the reliability constraint without task duplication, and iii) the Task Duplication Mapping approach (TDM), always performing task duplication without need to satisfy the reliability constraints. Table 2.b explores how the deadline constraint (D) affects the feasible and optimal solutions. For instance, when $0.8862 \leq D < 0.8978$, the proposed approach can find the solutions {S4-S5,S20}, where S4 is the optimal solution. RAM obtains the solutions {S4,S5}, with S4 the optimal solution. However, for TDM, only one solution is feasible, i.e., S20, while for more strict deadlines no solution

Table 2: Motivational Example. (*Optimal solutions are highlighted in bold).

Sol.	S1S	S2	S3	S4	S5	S6	S7	S8	S9	S10
f	f1	f2	f3	f4	f5	f1/f1	f1/f2	f1/f3	f1/f4	f1/f5
R	0.9753	0.9964	0.9994	0.9999	∼1	0.9994	0.9999	∼1	∼1	∼1
t	0.4994	0.4825	0.4677	0.4547	0.4431	0.9988	0.9818	0.9671	0.9541	0.9425
E	2.1169	2.7905	3.6959	4.926	6.6141	4.2338	4.9074	5.8128	7.0429	8.731
Sol.	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20
f	f2/f2	f2/f3	f2/f4	f2/f5	f3/f3	f3/f4	f3/f5	f4/f4	f4/f5	f5/f5
R	∼1	∼1	∼1	∼1	∼1	∼1	∼1	∼1	∼1	∼1
t	0.9649	0.9501	0.9372	0.9256	0.9353	0.9224	0.9108	0.9094	0.8978	0.8862
E	5.581	6.4864	7.7165	9.4046	7.3918	8.6219	10.31	9.852	11.5401	13.2282

a) Feasible solutions of three approaches under reliability constraint only

EFTMA	S5,S4,S20,S19,S18,S17,S16,S14,S15,S13,S10,S12,S9,S11,S8,S7
RAM	S5,S4
TDM	S20,S19,S18,S17,S16,S14,S15,S13,S10,S12,S9,S11,S8,S7

b) Feasible and **optimal*** solutions under reliability and deadline constraints

D	[0,0.4431)	[0.4431, 0.4547)	[0.4547, 0.8862)	[0.8862, 0.8978)	.....	[0.9818, -)
EFTMA	-	***S5***	S5,***S4***	S4-S5,S20/***S4***	.....	S4-S5,S7-S20/***S7***
RAM	**-**	***S5***	S5,***S4***	S5,***S4***	.....	S5,***S4***
TDM	**-**	**-**	**-**	***S20***	.....	S7-S20/***S7***

is found. Therefore, full replication approaches, such as TDM, are not able to find optimal solutions, due to strict deadline constraint, even for only a single task. For more relaxed deadlines, when $0.9818 \leq D$, the proposed approach can find the solutions {S4-S5,S7-S20}, where S7 is the optimal solution. TDM obtains the solutions {S7-S20}, with S7 the optimal solution. However, RAM obtains the solutions {S5,S4}, with S4 the optimal solution. Therefore, reliable original execution approaches, such as RAM, cannot exploit solutions due to task duplication, leading to higher energy consumption. On the contrary, the proposed EFTMA approach exploits the benefits of the aforementioned methods by applying partial task duplication, whenever is advantageous.

4 System Model

The following paragraphs present the models used in this work.

Task Model: We consider a set of N independent tasks, i.e., $\{\tau_1, \ldots, \tau_N\}$. Each task τ_i is measured in Worst Case Execution Cycles (WCEC) C_i. All tasks must be executed before a common deadline D, which is the scheduling period. No preemption occurs between different tasks executed on the same processor. Without loss of generality, in the rest of the paper, the release times of all tasks are considered at the start of the scheduling period. R_i^{th} denotes the reliability threshold of task τ_i.

Platform and Power Model: A multicore platform is considered with M homogeneous processors, i.e., $\{\theta_1, \ldots, \theta_M\}$. The multicore platform can support different DVFS schemes: i) task-level DVFS (TL-DVFS), where each task can be executed with its own frequency [7, 12, 23], ii) processor-level DVFS (PL-DVFS), where all tasks assigned on a processor have the same frequency [13], and iii) system level DVFS (SL-DVFS), where all system processors are assigned with the same frequency. For each core, there are L different Volt-

age/Frequency (V/F) pairs $\{(v_1, f_1), \ldots, (v_L, f_L)\}$. When task τ_i is assigned with frequency f_l, its execution time is calculated as $t_i = \frac{C_i}{f_l}$. For each processor θ_m, the power consumption is modeled as the sum of static power P_l^{sta} and dynamic power P_l^{dyn} [29], i.e., $P_l = P_l^{sta} + P_l^{dyn}$. The dynamic power consumption with V/F level (v_l, f_l) is given by $P_l^{dyn} = C_{eff} v_l^2 f_l$, where C_{eff} is the effective switching capacitance.

Fault Model and Reliability: We focus on soft errors that follow a Poisson Distribution with fault rate $\lambda(f)$ at frequency f, modeled as:

$$\lambda(f) = \lambda_0 \times 10^{d \frac{f_{\max} - f}{f_{\max} - f_{\min}}}, \tag{1}$$

where λ_0 is the average fault rate corresponding to the maximum frequency $f_{\max}$, d_0 is a positive constant, indicating the sensitivity of failure rate related to frequency scaling [4]. The reliability of a task τ_i, i.e., the probability of executing the task without any fault, follows the exponential model [4]. Hence, the reliability of a task at frequency f_l is given by

$$r_i(f_l) = e^{-\varphi_i(f_l)}, \tag{2}$$

where $\varphi_i(f_l) = \lambda(f_l) \times t_i$, with t_i the execution time of task τ_i [4, 5, 23].

If the reliability of task τ_i is larger than its reliability constraint, the execution is considered as reliable, and, thus, the reliability of τ_i is not modified, i.e., $R_i = r_i(f_l)$. Otherwise, the task τ_i is duplicated and the duplication task is executed on a different processor and the reliability of τ_i becomes:

$$R_i = 1 - [1 - r_i(f_l)][1 - r'_i(f_{l'})], \tag{3}$$

where $r'_i(f_{l'})$ is the reliability of duplication task of τ_i.

As our approach is applied offline to minimize energy consumption, we consider only task duplication, i.e., a single replica per task, in order not to unnecessarily increase the platform workload, in case no faults occur. An online mechanism can be applied, during execution, to further improve the energy consumption of our solution (by not executing replicas, when first executions are correct), and to deal with the low probability cases, where non-replicated tasks and both original and duplicated tasks are faulty, such as [30].

5 Energy-efficient Fault-tolerant Task Mapping Approach (EFTMA)

The goal is to minimize the total energy consumption of the system, subject to a set of reliability and real-time constraints. To achieve that, we decide: 1) frequency of original and duplicated tasks ($\boldsymbol{s}$ and $\boldsymbol{c}$); 2) duplication of original tasks ($\boldsymbol{\sigma}$); 3) allocation of original tasks ($\boldsymbol{q}$) and duplicated tasks ($\boldsymbol{d}$). Table 3 summarizes the notation of our formulation. The following paragraphs describe the constraints and objective function of EFTMA formulation, initially considering a multicore platform supporting TL-DVFS. Then, the formulation is extended for PL-DVFS and SL-DVFS schemes.

Parameters	
$\boldsymbol{M}$	$\{1,\ldots,M\}$, with M number of processors
$\boldsymbol{N}$	$\{1,\ldots,N\}$, with N number of tasks
$\boldsymbol{L}$	$\{1,\ldots,L\}$, with L number of voltage/frequency levels
C_i	WCEC of task τ_i
D	global deadline
(v_l, f_l)	the l^{th} voltage/frequency level
R_i^{th}	reliability threshold of task τ_i
Binary Variables	
$\sigma_i = 1$	if task τ_i is duplicated, else $\sigma_i = 0$
$q_{im} = 1$	if original task of τ_i executes on processor θ_m, else $q_{im} = 0$
$d_{im} = 1$	if duplication task of τ_i executes on processor θ_m, else $d_{im} = 0$
TL-DVFS:	$s_{il} = 1$, if original task of τ_i executes with f_l, else $s_{il} = 0$
	$c_{il} = 1$, if duplication task of τ_i executes with f_l, else $c_{il} = 0$
PL-DVFS:	$s_{ml} = 1$, if processor θ_m executes with f_l, else $s_{ml} = 0$
SL-DVFS:	$s_l = 1$, if system executes with f_l, else $s_l = 0$

Table 3: Main Notations

Task-Level DVFS scheme (TL-DVFS): We first describe the task mapping problem under TL-DVFS scheme as an MINLP, and then how to equivalently transform it into an MILP problem. Let $\mathbf{N} = \{1,\ldots,N\}$, $\mathbf{M} = \{1,\ldots,M\}$ and $\mathbf{L} = \{1,\ldots,L\}$.

i) Frequency assignment: Under TL-DVFS, the platform applies DVFS per task, and each task can only be assigned with one frequency level:

$$\sum\nolimits_{l\in\boldsymbol{L}} s_{il} = 1,\ \forall i \in \boldsymbol{N}, \tag{4}$$

$$\sum\nolimits_{l\in\boldsymbol{L}} c_{il} = \sigma_i,\ \forall i \in \boldsymbol{N}. \tag{5}$$

ii) Task duplication decision: The reliability of original task τ_i is expressed as $r_i = \sum_{l\in\boldsymbol{L}} s_{il} e^{-\varphi_i(f_l)}$, where $\varphi_i(f_l) = \lambda(f_l) \times T_i$ and $T_i = \sum_{l\in\boldsymbol{L}} s_{il} \frac{C_i}{f_l}$ (the execution time of original task τ_i when executed with frequency f_l). Let R_i^{th} denote the reliability threshold of task τ_i. If $0 < r_i \leq R_i^{th}$, the task needs to be duplicated, $\sigma_i = 1$, else (i.e., $r_i > R_i^{th}$), only the original task is executed, thus, $\sigma_i = 0$. In order to describe this behaviour, the following Lemma is introduced.

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 $x > y$, $c = 0$, we have $\delta - (1+\delta)c \leq x - y \leq 1 - c$, where δ is positive small value.*

Proof 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 C_1 and C_2 at the same time, 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, for the values of r_i, we have $r_i \in \{e^{-\varphi_i(f_1)}, \ldots, e^{-\varphi_i(f_L)}\}$. According to Lemma 1, the relationship between $R_i(f_l)$, R_i^{th} and σ_i is linearized as follows:

$$\delta - (1+\delta)\sigma_i \leq \sum\nolimits_{l \in \boldsymbol{L}} s_{il} e^{-\varphi_i(f_l)} - R_i^{th} \leq 1 - \sigma_i, \ \forall i \in \boldsymbol{N}. \tag{6}$$

iii) Task allocation: The original task, τ_i is executed on one processor:

$$\sum\nolimits_{m \in \boldsymbol{M}} q_{im} = 1, \ \forall i \in \boldsymbol{N}. \tag{7}$$

The original and its replica are allocated on different processors [12, 26]:

$$\sum\nolimits_{m \in \boldsymbol{M}} d_{im} = \sigma_i, \ \forall i \in \boldsymbol{N}, \tag{8}$$

$$q_{im} + d_{im} \leq 1, \ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}. \tag{9}$$

iv) Real-time constraints: Tasks (original and duplicated) assigned on processor θ_m should be executed within a common deadline D, thus

$$\sum\nolimits_{i \in \boldsymbol{N}} q_{im} T_i + \sum\nolimits_{i \in \boldsymbol{N}} d_{im} t_i \leq D, \ \forall m \in \boldsymbol{M}, \tag{10}$$

where $T_i = \sum_{l \in \boldsymbol{L}} s_{il} \frac{C_i}{f_l}$ and $t_i = \sum_{l \in \boldsymbol{L}} c_{il} \frac{C_i}{f_l}$ are the execution times of original and duplicated task of τ_i, respectively.
v) Objective Function: The energy consumption of original task τ_i and its potential duplicated task is:

$$E_i = \sum\nolimits_{l \in \boldsymbol{L}} s_{il} P_l \frac{C_i}{f_l} + \sum\nolimits_{l \in \boldsymbol{L}} c_{il} P_l \frac{C_i}{f_l} \tag{11}$$

Based on the objective function and the aforementioned constraints, the Primal Problem (**PP**) is formulated as

$$\mathbf{PP}: \min_{\boldsymbol{s},\boldsymbol{c},\boldsymbol{q},\boldsymbol{\sigma},\boldsymbol{d}} E_s \tag{12}$$
$$\text{s.t.} \begin{cases} (4), (5), (6), (7), (8), (9), (10) \\ s_{il}, c_{il}, q_{im}, \sigma_i, d_{im} \in \{0,1\}, \ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}, \ \forall l \in \boldsymbol{L}. \end{cases}$$

where $E_s = \sum_{i \in \boldsymbol{N}} E_i$ is the total energy consumed by all tasks until the deadline, according to Equation (11).

Since the nonlinear items exist in Equation (10) (i.e., $q_{im}T_i$ and $d_{im}t_i$), **PP** is an INLP problem, which is difficult to solve optimally. In order to find the optimal solution, as well as to simplify the structure of the problem, we equivalently transform **PP** to an MILP problem. By applying *variable replacement* method, the nonlinear variable combinations are replaced equivalently by an MILP formulation. First, we observe that $s_{il} \in \{0,1\}$ and C_i is large enough, one cycle has a negligible impact on the solution. Thus, T_i and t_i can be relaxed to the continuous variables: $0 \leq T_i = \sum_{l \in \boldsymbol{L}} s_{il} \frac{C_i}{f_l} \leq \overline{T_i}$ and $0 \leq t_i = \sum_{l \in \boldsymbol{L}} c_{il} \frac{C_i}{f_l} \leq \overline{T_i}$, where $\overline{T_i} = \frac{C_i}{f_{min}}$. To linearize $q_{im}T_i$ and $d_{im}t_i$, we introduce the following lemma:

Lemma 2 *Given two positive constants s_1 and s_2, there are two constraint spaces $P_1 = \{[t,b,x]|t = bx, -s_1 \leq x \leq s_2, b \in \{0,1\}\}$ and $P_2 = \{[t,b,x]| - bs_1 \leq t \leq bs_2, t + bs_1 - x - s_1 \leq 0, t - bs_2 - x + s_2 \geq 0, b \in \{0,1\}\}$, then $P_1 \rightleftharpoons P_2$.*

Proof i) $P_1 \rightharpoonup P_2$: According to $t = bx$ and $-s_1 \leq x \leq s_2$, then $-bs_1 \leq t \leq bs_2$. Based on $-s_1 \leq x \leq s_2$ and $b \in \{0,1\}$, then $(b-1)(x-s_2) \geq 0$ and $(b-1)(x+s_1) \leq 0$. Therefore, $t - bs_2 - x + s_2 \geq 0$ and $t + bs_1 - x - s_1 \leq 0$ hold. ii) $P_1 \leftharpoonup P_2$: When $b = 0$, $t = 0$ and $-s_1 \leq x \leq s_2$ according to P_2 space definition. When $b = 1$, then $-s_1 \leq t = x \leq s_2$ from P_2. Thus, $P_1 \rightleftharpoons P_2$.

Based on Lemma 2, the continuous variables $\alpha_{im} = q_{im}T_i$ and $\beta_{im} = d_{im}t_i$ are introduced, and Equation (10) can be replaced by:

$$\sum\nolimits_{i\in\boldsymbol{N}} \alpha_{im} + \sum\nolimits_{i\in\boldsymbol{N}} \beta_{im} \leq D,\ \forall m \in \boldsymbol{M}, \tag{13a}$$

$$-\overline{T_i}q_{im} + \alpha_{im} \leq 0,\ -T_i + \alpha_{im} \leq 0,\ \overline{T_i}q_{im} + T_i - \alpha_{im} \leq \overline{T_i}, \quad \forall i \in \boldsymbol{N},\ \forall m \in \boldsymbol{M}, \tag{13b}$$

$$-\overline{T_i}d_{im} + \beta_{im} \leq 0,\ -t_i + \beta_{im} \leq 0,\ \overline{T_i}d_{im} + t_i - \beta_{im} \leq \overline{T_i}, \quad \forall i \in \boldsymbol{N},\ \forall m \in \boldsymbol{M}. \tag{13c}$$

Therefore, the primal problem (12) is equally reformulated as follows:

$$\textbf{EFTMA-TL}: \min_{\substack{\boldsymbol{s},\boldsymbol{c},\boldsymbol{q},\boldsymbol{\sigma},\boldsymbol{d},\boldsymbol{h},\\ \boldsymbol{t},\boldsymbol{\chi},\boldsymbol{\xi}}} \sum\nolimits_{i\in\boldsymbol{N}} \sum\nolimits_{l\in\boldsymbol{L}} P_l \frac{C_i}{f_l}(s_{il} + c_{il}) \tag{14}$$

$$\text{s.t.} \begin{cases} (4) \sim (9), (13a), (13b), (13c), , \\ -\sum\limits_{l\in\boldsymbol{L}} \dfrac{C_i}{f_l} s_{il} + T_i = 0,\ \forall i \in \boldsymbol{N}, \\ -\sum\limits_{l\in\boldsymbol{L}} \dfrac{C_i}{f_l} c_{il} + t_i = 0,\ \forall i \in \boldsymbol{N}, \\ s_{il}, c_{il}, q_{im}, \sigma_i, d_{im} \in \{0,1\}. \\ 0 \leq T_i, t_i, \alpha_{im}, \beta_{im} \leq \overline{T_i},\ \forall i \in \boldsymbol{N},\ \forall m \in \boldsymbol{M},\ \forall l \in \boldsymbol{L}. \end{cases}$$

Since all the variables (binary and continuous) are coupled linearly with each other, **PP1** is an MILP problem.

Processor-Level DVFS scheme (PL-DVFS): When the platform supports PL-DVFS, each processor can have a single frequency level and the tasks assigned to the processor are executed with same frequency:

$$\sum\nolimits_{l\in\mathcal{L}} s_{ml} = 1,\ \forall m \in \mathcal{M}, \tag{15}$$

For task τ_i, the frequency is decided together with task-to-processor allocation (q_{im}) and frequency-to-processor assignment (s_{ml}), as $\sum_{m\in\mathcal{M}} q_{im}(\sum_{l\in\mathcal{L}} s_{ml})f_l$. Then, the reliability is $R_i = \sum_{m\in\mathcal{M}} q_{im}(\sum_{l\in\mathcal{L}} s_{ml})e^{-\varphi_i(f_l)}$. The execution time of the original and duplicated task of τ_i (if duplication takes place) are $T_i = \sum_{m\in\mathcal{M}} q_{im}(\sum_{l\in\mathcal{L}} s_{ml})\frac{C_i}{f_l}$ and $t_i = \sum_{m\in\mathcal{M}} d_{im}(\sum_{l\in\mathcal{L}} s_{ml})\frac{C_i}{f_l}$, respectively. These equations replace the corresponding equations of EFTMA-TL, and the proposed approach under PL-DVFS scheme is formulated as

$$\textbf{EFTMA-PL}: \min_{\boldsymbol{s},\boldsymbol{q},\boldsymbol{d},\boldsymbol{\sigma}} \sum\nolimits_{i\in\boldsymbol{N}} \left[\sum\nolimits_{m\in\boldsymbol{M}} q_{im} \left(\sum\nolimits_{l\in\boldsymbol{L}} s_{ml} \frac{C_i}{f_l} P_l \right) + \sum\nolimits_{m\in\boldsymbol{M}} d_{im} \left(\sum\nolimits_{l\in\boldsymbol{L}} s_{ml} \frac{C_i}{f_l} P_l \right) \right]$$
$$\text{s.t.} \begin{cases} (6) \sim (10), (15) \\ s_{ml}, q_{im}, d_{im}, \sigma_i \in \{0,1\},\ \forall i \in \boldsymbol{N},\ \forall m \in \boldsymbol{M},\ \forall l \in \boldsymbol{L}. \end{cases}$$

System-Level DVFS scheme (SL-DVFS): When the platform supports SL-DVFS, all processors are assigned with same frequency:

$$\sum\nolimits_{l\in\mathcal{L}} s_l = 1. \tag{16}$$

For all tasks, the frequency is $\sum_{l\in\boldsymbol{L}} s_l f_l$, the reliability is $R_i = \sum_{l\in\boldsymbol{L}} s_l e^{-\varphi_i(f_l)}$ and the execution time of original and duplicated task (if duplication takes place) is $\sum_{l\in\boldsymbol{L}} s_l \frac{C_i}{f_l}$. Then, the proposed SL-DVFS approach is formulated as

$$\textbf{EFTMA-SL}: \min_{\boldsymbol{s},\boldsymbol{q},\boldsymbol{d},\boldsymbol{\sigma}} \sum\nolimits_{i\in\boldsymbol{N}} \left(\sum\nolimits_{l\in\boldsymbol{L}} s_l \frac{C_i}{f_l} P_l \right) + \sum\nolimits_{i\in\boldsymbol{N}} \left(\sum\nolimits_{l\in\boldsymbol{L}} \sigma_i s_l \frac{C_i}{f_l} P_l \right)$$
$$\text{s.t.} \begin{cases} (7) \sim (10), (16) \\ s_l, q_{im}, d_{im}, \sigma_i \in \{0,1\},\ \forall i \in \boldsymbol{N},\ \forall m \in \boldsymbol{M},\ \forall l \in \boldsymbol{L}. \end{cases}$$

Similarly, the problem formulations of EFTMA-PL and EFTMA-SL are safely and equivalently transformed into MILP forms, using the same method with TL-DVFS.

6 Experimental Result

The aim of this work is to compare the proposed task mapping approach with existing task mapping approaches and also explore the impact different DVFS schemes may have in the task mapping decisions. Current platforms usually support one DVFS scheme only. Performing experiments using different platforms, having different characteristics and power consumption, does not help at this initial exploration stage. The experiments we present are thus based on simulations and in each experimental set-up the same platform and power

model is used. It makes it possible to compare the quality of the obtained results among TL-DVFS, PL-DVFS and SL-DVFS schemes.

Experimental set-up: The processor model we used in the experiments is based on RISC-V Instruction Set Architecture (ISA). We have used a high-level C++ model with 32-bit RISC-V ISA and standard 5-stage pipeline, i.e., the open-source Comet processor [31]. Regarding the DVFS, $L = 6$ voltage/frequency levels are used, based on the work of [32] considering 64 nm technology, as depicted in Table 4. To obtain realistic inputs for our experiments, we count the execution cycles and Memory Accesses (MA) of common benchmarks from MiBench suite [33], using Comet simulator. The sources of timing variability are eliminated to obtain safe and context-independent measurement [34] without interferences (WCEC_{iso}). Then, the WCEC_{inf}, considering worst case interferences from the other processors, is computed. As the contribution of this paper is not WCET estimation, a trivial pessimistic approach is applied: all processors may conflict during a memory access. Thus, the interference cost is given by (M-1)*MA*Main_Memory_Access_Delay (Table 4).

The proposed approach (EFTMA) is compared to two SoA approaches: i) the Reliability-Aware Mapping (RAM) approach, similar to [10] and "ESRG" algorithm in [12], and ii) the Duplication Mapping approach (TDM), always performing task duplication , similar to [7, 12], when the number of replicas is two, or to [27], with 100% task duplication. RAM is the typical way to meet the required reliability, without replication. When task replication is applied, 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: i) the number of processors ($M = 2, 4$), ii) the size of the task set ($N = 10, 20$), iii) the platform DVFS scheme (from flexible TL-DVFS to more restricted PL-DVFS and SL-DVFS), and iv) the global deadline D (from strict D to more relaxed ones, using $D = k \times \frac{N}{M} \times \frac{1}{2}(\frac{C_{max}}{f_{min}} + \frac{C_{max}}{f_{max}})$, with a step of 0.1 for $N = 10$ and 0.2 for $N = 20$ by adjusting k). For each set-up, 20

Table 4: Platform and benchmark characteristics

$\boldsymbol{v_l}$ (V)	0.85	0.90	0.95	1.00	1.05	1.1
$\boldsymbol{f_l}$ (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**	$\boldsymbol{WCEC_{iso}}$	$\boldsymbol{WCEC_{inf}}$	
			$M = 2$	$M = 4$
`matmul (int)`	371,957	3,313,958	77,705,358	226,488,158
`matmul (int64)`	507,133	4,055,289	78,446,689	308,335,089
`qsort (int)`	184,089	875,616	75,267,016	111,329,016
`qsort (int64)`	259,553	1,219,854	75,611,254	156,951,654
`qsort (float)`	185,437	1,745,122	76,136,522	113,007,322
`dijkstra`	117,151	766,369	75,157,769	71,056,969
`blowfish`	110,330	3,058,991	77,450,391	69,256,991
`stringsearch`	597,608	13,093,544	87,484,944	371,658,344

experiments are performed. For each experiment, the WCEC C_i of each task is selected within the range $[1 \times 10^8, 4 \times 10^8]$, based on WCEC_{inf} (Table 4), including the time overhead for DVFS change (e.g., 10–150 μs [21,22]) and the sanity checks at the end of a task [7]. The reliability threshold R_i^{th} is selected within the range $[0.9990, 0.9995]$, considering a typical magnitude 10^{-3} for reliability target [7]. Such a reliability target for a task is inline with safety standards, such as ISO 26262 for automotive systems, DO-178B for avionics systems and IEC 61508 for industrial software systems [10,12]. The used fault model has an average failure rate $\lambda_0 = 5 \times 10^{-5}$ faults/sec and $d_0 = 3$ [4].

Note that the above numbers provide only specific values to problem parameters for experiments, without affecting the problem structure. The approaches are implemented and solved with Gurobi 9.0.2 (MILP solver) on several servers, as hundreds of experiments took place.

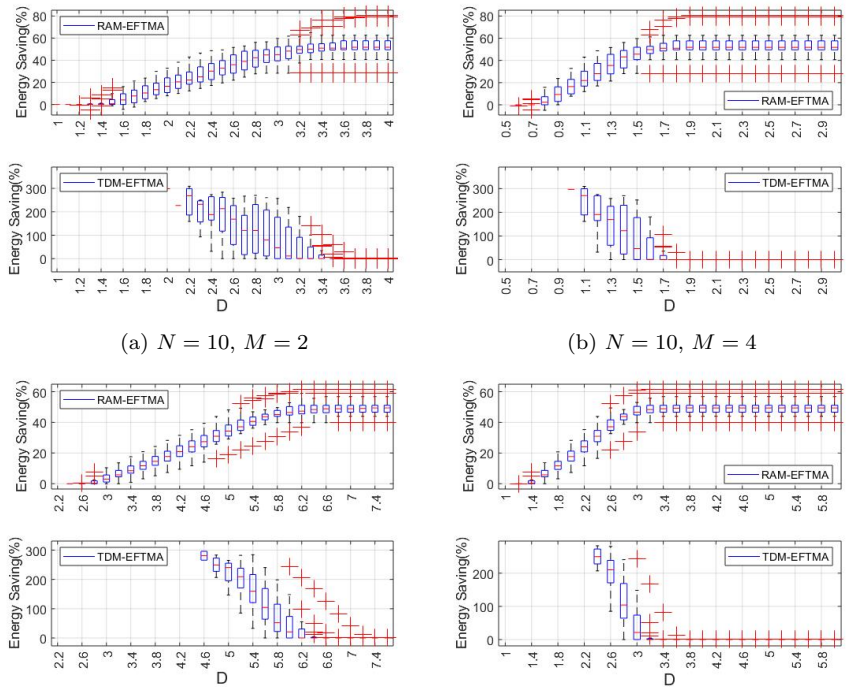
To evaluate the behavior of the proposed approach, we compute:

1. Energy savings, i.e., the gain in energy consumption that EFTMA achieved compared to RAM and TDM.
2. Reliability, i.e., the achieved reliability in each set-up, for all approaches.
3. Feasibility, i.e., percentage of experiments that found the optimal solution among the 20 experiments of each set-up.
4. Task duplication, i.e., the average percentage of tasks that EFTMA approach decided to duplicate among the 20 experiments, for each set-up.
5. Solution time, i.e., the average time required for all approaches to find a solution among the 20 experiments, for each set-up.

Energy savings: The gain in energy consumption, achieved by the proposed approach compared to RAM and TDM, is depicted in Fig. 1 (TL-DVFS), Fig. 2 (PL-DVFS) and Fig. 3 (SL-DVFS). Notice that, when points do not appear in the figures, the corresponding approach found no solutions. Table 5 depicts the achieved minimum, average and maximum gains compared to RAM and TDM approaches. The minimum gain is zero (or close to) for strict deadlines for RAM, since EFTMA behaves as RAM in this case, executing reliably only the original tasks, i.e. with no duplication. For relaxed deadlines, EFTMA behaves as TDM, duplicating all tasks.

Table 5: min., avg. and max. energy saving gains (%)

		TL-DVFS			**PL-DVFS**			**SL-DVFS**		
N	**M**	**Min.**	**Avg.**	**Max**	**Min.**	**Avg.**	**Max**	**Min.**	**Avg.**	**Max**
RAM - EFTMA										
10	2	0	29.8	53.9	0	17.2	59.6	0	29.5	89.4
10	4	0	40.8	53.9	0	37.0	53.9	0	54.9	89.4
20	2	0	31.2	49.7	0	16.9	54.1	0	29.9	81.1
20	4	0	38.8	49.7	0	35.4	49.7	0	52.3	81.1
TDM - EFTMA										
10	2	0.1	103.9	298.7	0	77.1	229.8	0	61.8	180.0
10	4	0.1	69.8	296.6	0.1	51.8	295.6	0	34.1	180.0
20	2	0.01	87.3	280.9	0	69.7	258.4	0	55.5	159.8
20	4	0.01	32.9	248.1	0.1	29.6	232.7	0	21.4	157.1



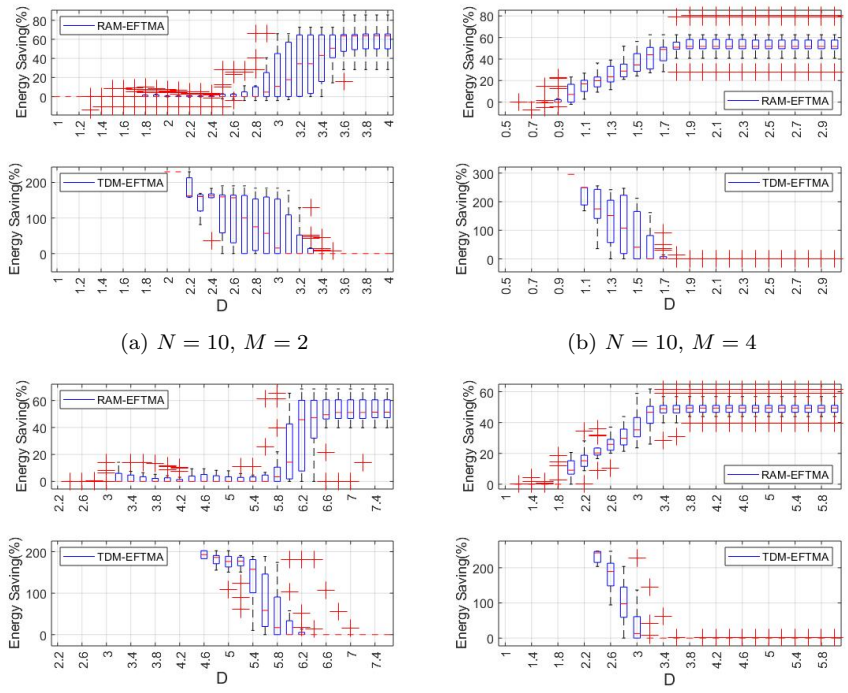
(a) $N = 10$, $M = 2$ (b) $N = 10$, $M = 4$

(c) $N = 20$, $M = 2$ (d) $N = 20$, $M = 4$

Fig. 1: TL-DVFS: Energy Savings

Number of processors: When the task set size is the same and the number of processors increases, we observe that EFTMA, compared to RAM, has lower energy consumption and the average energy savings are increasing. For instance, on average, gains change from 29.8% to 40.8% (TL-DVFS), 17.2% to 37.0% (PL-DVFS) and 29.5% to 54.9% (SL-DVFS), when M increases from 2 to 4 with $N = 10$. In fact, with processor number increasing, the proposed approach has more available resources, thus it can duplicate more tasks and execute them with low frequency and guarantee the reliability requirements. Energy consumption is thus reduced. On the contrary, RAM should always meet the reliability constraint. Although it uses more processors to execute in parallel the tasks, it cannot reduce their energy consumption, as EFTMA can do, due to the fact that the reliability constraints have to be satisfied using only original tasks (no duplication).

Compared to TDM, the energy savings of the proposed approach are reduced when the processor number increases. For instance, on average, gains change from 103.9% to 69.8% (TL-DVFS), 77.1% to 51.8% (PL-DVFS) and 61.8% to 34.1% (SL-DVFS), when M increases from 2 to 4 with $N = 10$. Actually, with more processors or with more relaxed deadline, the proposed approach and TDM tend to behave similarly, i.e., duplicating all tasks. How-



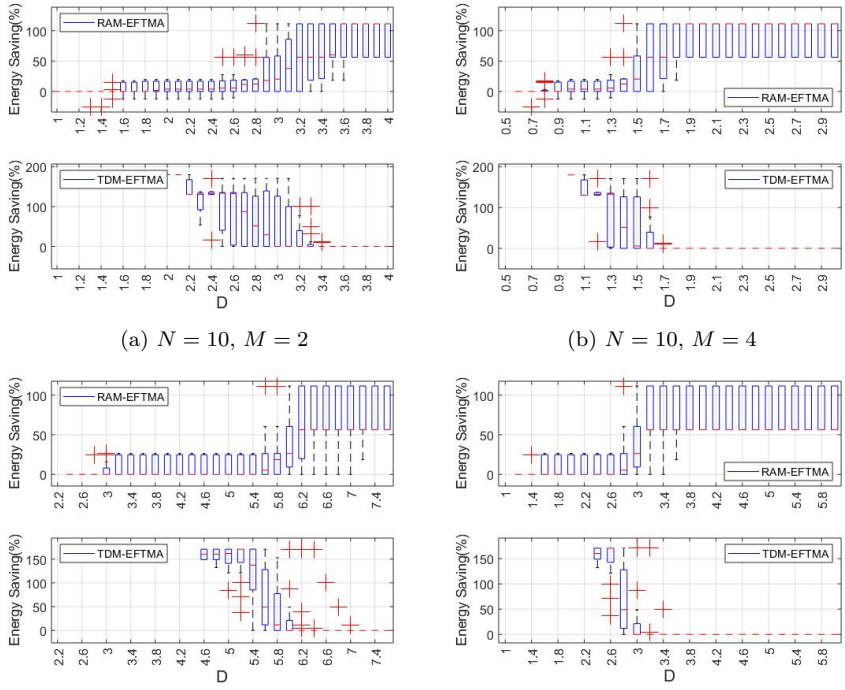
(a) $N = 10$, $M = 2$ (b) $N = 10$, $M = 4$

(c) $N = 20$, $M = 2$ (d) $N = 20$, $M = 4$

Fig. 2: PL-DVFS: Energy Saving

ever, with less processors, EFTMA can execute more tasks using only the original copy, while TDM has to duplicate all tasks, thus increasing the energy consumption.

Number of tasks: As expected, with task number increasing, more energy is consumed in all approaches, under all DVFS schemes. When the number of tasks is increased from $N = 10$ to $N = 20$, with the same number of processors, the energy savings remain high for the proposed approach, under all three DVFS schemes. With task number increasing and a given number of processors, it takes a wider deadline region for the energy saving gains to become stable. For instance, for $N = 10$ and TL-DVFS, during the deadline region $D = [1, 3.7]$ (top Fig. 1a), the energy gain of EFTMA, compared to RAM, increases with the deadline becoming more relaxed. After $D = 3.7$, the energy gain becomes stable. For $N = 20$ with same number of processors (Fig. 1c), the energy gain increases with increasing deadline during a wider region $D = [2.6, 6.6]$, and after $D = 6.6$, it stays fixed. A similar behavior is observed for EFTMA, compared to TDM, but with energy saving gain decreasing. Actually, with more tasks, a larger time is required for their execution, until more relaxed deadlines where energy saving gains become stable.



(a) $N = 10$, $M = 2$ (b) $N = 10$, $M = 4$

(c) $N = 20$, $M = 2$ (d) $N = 20$, $M = 4$

Fig. 3: SL-DVFS: Energy Saving

DVFS flexibility: Based on Table 5, a general observation is that TL-DVFS scheme achieves promising energy savings for EFTMA. Actually the frequency assignment in TL-DVFS is the most flexible one, since it is performed per task. PL-DVFS comes next, as it performs frequency assignment per processor. SL-DVFS is the least flexible DVFS scheme, since all tasks are assigned the same frequency. Comparing the energy gains of TL-DVFS and PL-DVFS schemes, the average energy gains, between EFTMA and RAM, are decreased, e.g., from 40.8% to 37.0% and 38.8% to 35.4%, for $N = 10$ and $N = 20$, when $M = 4$. Comparing to SL-DVFS, the average energy savings are increased, especially when the number of processors in the platform becomes larger, e.g., from 40.8% to 54.9% and from 38.8% to 52.3%, for $N = 10$ and $N = 20$, when $M = 4$. With the most flexible DVFS scheme, i.e., TL-DVFS, RAM performs a more fine-grained frequency assignment, achieving a lower energy consumption, compared to SL-DVFS. With SL-DVFS, RAM is obliged to select a high frequency, in order to meet the highest reliability threshold among the tasks, and thus, all tasks must be executed with this high frequency, leading to large energy consumption. On the contrary, EFTMA is able to better exploit frequency assignement, even for the less flexible SL-DVFS scheme. Regarding TDM, comparing the average energy gains between TL-DVFS and PL-DVFS

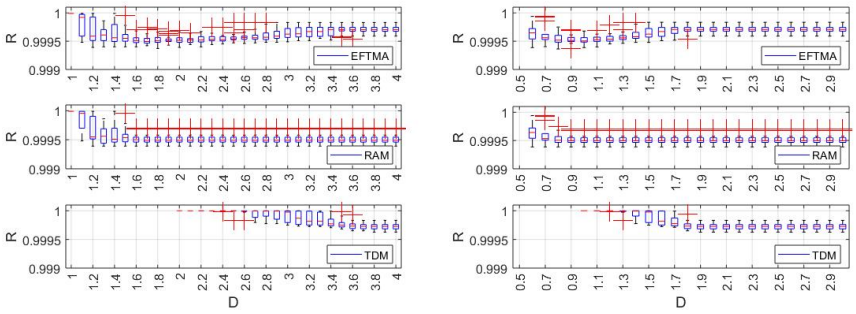
and SL-DVFS schemes, we observe a decrease, e.g., from 103.9% (TL-DVFS) to 77.1% (PL-DVFS) and to 61.8% (SL-DVFS), for $N = 10$ and $M = 2$, as the flexibility in deciding frequencies is reduced.

Deadline restriction: From Fig. 1, Fig. 2 and Fig. 3, we observe that the energy gains, between the proposed approach and RAM, remain small at strict deadlines. For instance, for $N = 20$ and $M = 2$, the energy gain is smaller than 10% for $D = 2.2$ to $D = 3.4$ in TL-DVFS (Fig. 1c) and smaller than 5% for $D = 2.2$ to $D = 5.8$ in PL-DVFS (Fig. 2c), and for $D = 2.2$ and $D = 5.6$ in SL-DVFS (Fig. 3c). When the deadline is strict, the proposed approach behaves as RAM: there is no available time slack, and thus, EFTMA assigns high frequencies without applying task duplication. At less strict deadlines, the proposed approach explores any available time slack to duplicate tasks. For instance, for $N = 20$ and $M = 2$, the energy gain reaches its maximum for $D = 6.2$ to $D = 7.6$ in TL-DVFS (Fig. 1c), $D = 6.6$ to $D = 7.6$ in PL-DVFS (Fig. 2c), and $D = 6.2$ to $D = 7.6$ in SL-DVFS (Fig. 3c). The trend is inverted between the proposed approach and TDM. When TDM can find solutions, the energy gain reaches its maximum at strict deadlines, since the proposed approach can execute original tasks at a high frequency, while TDM requires duplication of all tasks. The minimum (0% for most cases) is observed at quite relaxed deadlines, since in this case EFTMA and TDM behave similarly.

Reliability Achievement: Fig. 4, Fig. 5 and Fig. 6 show the reliability achievements of all approaches and DVFS schemes.

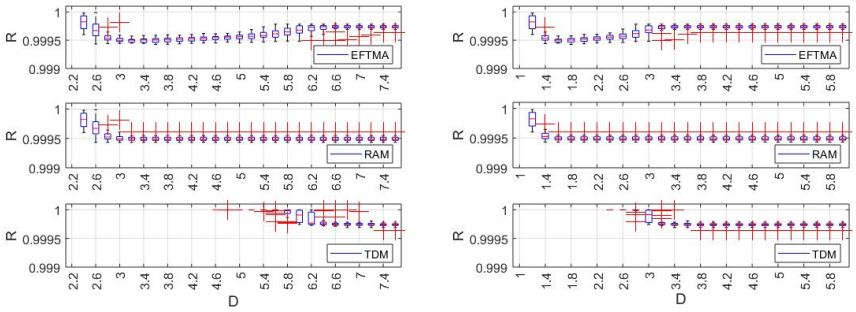
Regarding EFTMA, it achieves higher reliability than RAM, except in very strict deadlines. Compared to TDM, EFTMA provides lower reliability for tight deadlines, as it partially duplicates the task-set. However, as discussed in next section, EFTMA can find solutions when TDM cannot. When the deadline is not so strict, e.g., for $D = 1.9$ to $D = 3$ in TL-DVFS (Fig. 4b) and PL-DVFS (Fig. 5b), EFTMA achieves the same reliability as TDM, since they behave in a similar way.

Regarding RAM, it has the lowest reliability in TL-DVFS, without violating the reliability constraints. This is because RAM has as requirement to meet the reliability threshold. However, for the less flexible in frequency assignment PL-DVFS and SL-DVFS schemes, RAM is obliged to select a higher frequency, even for tasks with lower reliability threshold. As a result, the achieved reliability of RAM is increased, especially in SL-DVFS scheme where all tasks are executed with same frequency. This can be observed that in Fig. 6 (SL-DVFS), RAM generally achieves higher reliability than in Fig. 4 (TL-DVFS) and Fig. 5 (PL-DVFS). Another observation is that the proposed approach and RAM obtain same reliability at strict deadlines, e.g., when $D = 1$ to $D = 1.5$ in TL-DVFS (Fig. 4a) and $D = 1$ to $D = 1.4$ in PL-DVFS (Fig. 5a), because in strict deadlines EFTMA behaves as RAM. Regarding TDM, when it can find a solution, it provides a high reliability, since it duplicates all tasks. Therefore, changing from TL-DVFS to PL-DVFS and SL-DVFS, has a low impact on the achieved reliability. When a solution is found in strict deadlines, it has usually high reliability, but at the price of high energy consumption, due to the high frequencies required to meet the strict deadlines. For all DVFS schemes



(a) $N = 10$, $M = 2$

(b) $N = 10$, $M = 4$



(c) $N = 20$, $M = 2$

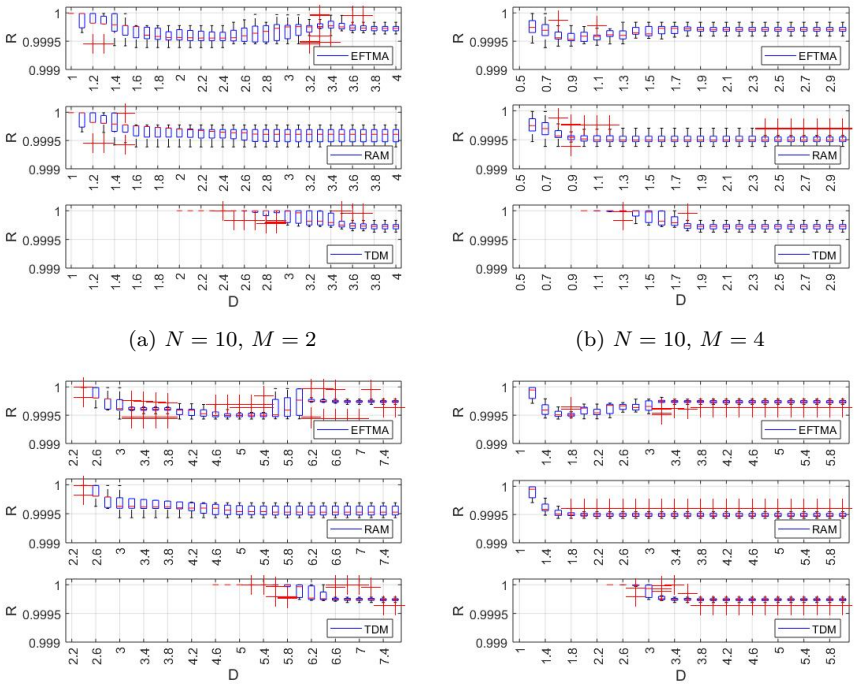
(d) $N = 20$, $M = 4$

Fig. 4: TL-DVFS: Reliability Achievement.

in relaxed deadlines, the reliability achieved by the proposed approach and TDM is the same, since EFTMA and TDM behave similarly, when deadlines are relaxed enough.

Feasibility: Fig. 7 depicts the feasibility of TL-DVFS scheme for the 3 approaches. Note that we observed no real difference in the feasibility behavior among TL-DVFS, PL-DVFS and SL-DVFS schemes in the performed experiments so only TL-DVFS feasibility is shown.

Comparing EFTMA and TDM feasibilities, EFTMA can find solutions in significantly more experiments, because EFTMA is not obliged to duplicate every task, whereas TDM does. More precisely, when feasibility has not reached 100%, for both approaches, EFTMA finds a solution, on average, in 63.5% (Fig. 7a), 60.5% (Fig. 7b), 65.26% (Fig. 7c) and 61.5% (Fig. 7d) more experiments than TDM. We also observe that TDM finds solutions only after the deadline $D = 1.9$ (Fig. 7a), $D = 0.9$ (Fig. 7b), $D = 4.4$ (Fig. 7c) and $D = 2.2$ (Fig. 7d). Moreover, EFTMA achieves 100% feasibility in earlier deadlines than TDM, i.e., $D = 1.5$ (Fig. 7a), $D = 0.8$ (Fig. 7b), $D = 3$ (Fig. 7c) and $D = 1.6$ (Fig. 7d). With increasing number of processors (comparing left and right parts in), the capability of TDM to find solutions improves, as more pro-



(a) $N = 10$, $M = 2$ (b) $N = 10$, $M = 4$

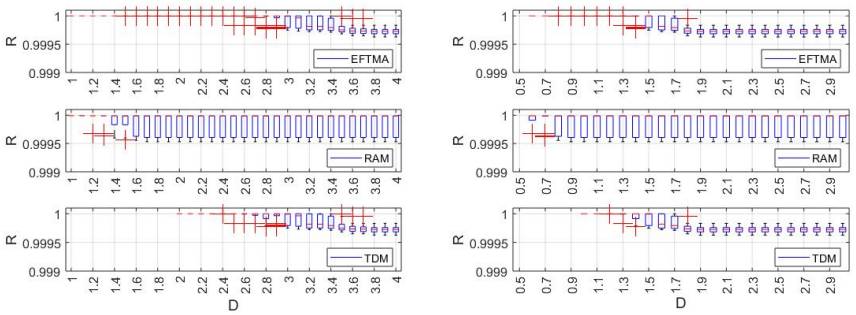
(c) $N = 20$, $M = 2$ (d) $N = 20$, $M = 4$

Fig. 5: PL-DVFS: Reliability Achievement.

cessors are available to schedule original and duplication tasks. Furthermore, all approaches achieve 100% feasibility at earlier deadlines.

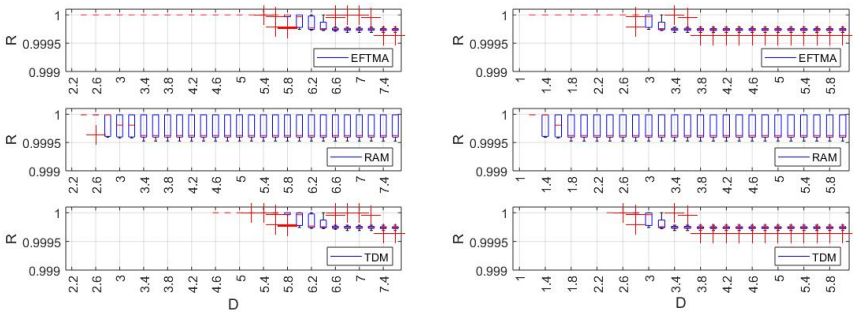
Comparing EFTMA with RAM feasibilities, they have the same feasibility, due to the values of reliability thresholds, i.e., the reliability thresholds can be achieved by executing the original task with a high processor frequency.

Task duplication: Fig. 8 depicts the percentage of duplicated tasks by EFTMA in all DVFS schemes. RAM approach does not duplicate tasks (0%) and TDM duplicates all tasks (100%). Under same set-up, except the cases of very strict and very relaxed deadlines, EFTMA decides the highest task duplication in TL-DVFS, and the least in SL-DVFS. The reasons are the flexibility in assigning task frequencies in different DVFS schemes and the reliability threshold's value. TL-DVFS assigns frequencies per task, providing more opportunities in duplicating task and executing them with low frequency, in order to achieve energy savings. In SL-DVFS, all tasks are executed with same frequency, and thus, the task frequency assignment is restricted. This reduces the possibilities to reduce energy consumption, by task duplication and execution with lower energy consumption. At strict deadlines, for all DVFS schemes, few tasks are duplicated. This is because the execution of duplicated tasks requires time and resources. Thus, it is not possible to duplicate tasks and meet the strict



(a) $N = 10$, $M = 2$

(b) $N = 10$, $M = 4$



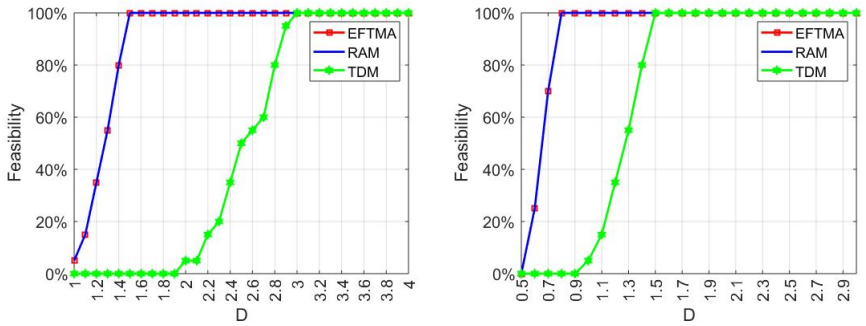
(c) $N = 20$, $M = 2$

(d) $N = 20$, $M = 4$

Fig. 6: SL-DVFS: Reliability Achievement.

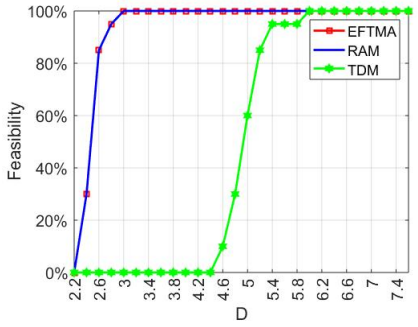
deadlines. When the number of processors is increased, more tasks can be duplicated. When the deadline is more relaxed, EFTMA take advantage of the time slack and duplicates more tasks, using lower frequency, and thus, achieving less energy consumption. At very relaxed deadlines, the percentage of task duplication does not always reach 100% for TL-DVFS and PL-DVFS, as SL-DVFS does. This occurs when the reliability threshold of a task is satisfied, by executing only the original task and, at same time, its energy consumption is lower than the energy consumption, when the task is duplicated. When TL-DVFS and PL-DVFS schemes are used, the proposed approach has flexibility in assigning different frequencies among tasks. Therefore, task duplication does not occur for this task, under TL-DVFS and PL-DVFS schemes. However, in SL-DVFS, all task are executed with same frequency. Thus, EFTMA applies task duplication to enable a lower system frequency, leading to less energy consumption.

Solution Time: Tables 6, 7, 8 and 9 provide the average time required to find the solution, for all approaches and DVFS schemes. For the proposed approach, when deadline is strict or relaxed, less time is needed to obtain the solutions. However, for intermediate deadlines, more time is required, since EFTMA explores the available time slack to decide which, and how many,

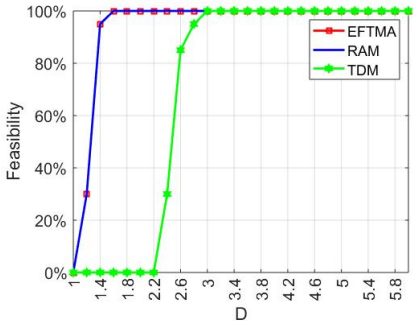


(a) $N = 10$, $M = 2$

(b) $N = 10$, $M = 4$



(c) $N = 20$, $M = 2$



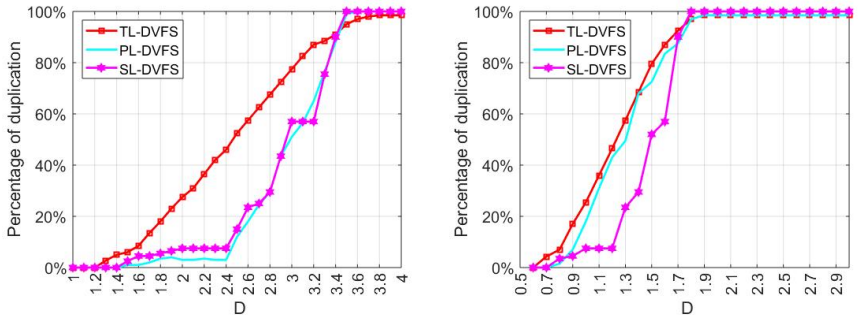
(d) $N = 20$, $M = 4$

Fig. 7: Feasibility of three approaches under TL-, PL-and SL-DVFS schemes.

tasks to be duplicated, without violating constraints, while consuming the least energy. TDM is the most time expensive approach, because all tasks are duplicated, increasing the number of tasks to be scheduled, and thus, the time to find the solutions. For RAM, as it only executes original tasks, it takes the least time to obtain a solution. However, it provides less energy savings as we observed. With same number of tasks and different number of processors (Table 6 vs. Table 7 and Table 8 vs. Table 9), more time is required; with more resources, the solver requires more time to optimally decide how to use the extra processors in order to obtain the global optimal solution.

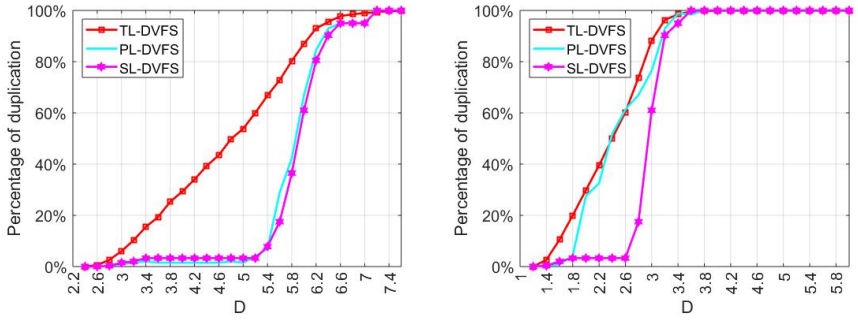
7 Conclusion, limitations and future work

In this paper, an energy-aware fault tolerant task mapping approach is proposed to minimize the total energy consumption, while satisfying the real-time and reliability constraints, by simultaneously determining task allocation, task duplication and frequency assignment, under three types of DVFS schemes. The proposed approach is compared with a reliability-aware task mapping approach and a fault-tolerant task mapping approach from state-of-the-art,



(a) $N = 10$, $M = 2$

(b) $N = 10$, $M = 4$



(c) $N = 20, M = 2$

(d) $N = 20$, $M = 4$

Fig. 8: Task Duplication (EFTMA) under all DVFS schemes.

under several scenarios. The experiment results show our approach is able to provide better energy savings, and it is able to find solutions, even when existing approaches may fail.

The proposed approach focuses on multicore platforms with homogeneous cores and independent tasks. In our future work, we will enhance the proposed approach to heterogeneous platforms, by considering execution efficiency of a core when it executes a given task and inserting additional constraints (e.g., task precedence and task non-overlapping constraints) into the task mapping problem, so as to be able to handle dependent task graphs. Furthermore, since task mapping on multicore platforms is a NP-hard problem, obtaining an optimal solution in a short time with the proposed approach when the problem size is large is not realistic enough. Therefore, our next step is to design controllable heuristics for EFTMA, in order to reduce the time required to find a solution and thus enhance the scalability of the proposed approach. The results presented in this paper can be used to characterise the quality of such heuristic approaches.

Table 6: Solution time (seconds) when $N = 10$, $M = 2$ for all DVFS schemes.

	TL-DVFS			PL-DVFS			SL-DVFS		
D	EFTMA	RAM	TDM	EFTMA	RAM	TDM	EFTMA	RAM	TDM
1.0	0.11	0.06	-	2.48	0.18	-	0.38	0.08	-
1.1	3.81	1.62	-	1.67	0.15	-	0.41	0.09	-
1.2	1.23	0.50	-	1.07	0.11	-	0.29	0.16	-
1.3	1.77	0.08	-	1.27	0.16	-	0.45	0.13	-
1.4	0.82	0.03	-	1.23	0.10	-	0.36	0.14	-
1.5	1.96	0.62	-	1.13	0.09	-	0.28	0.14	-
1.6	0.53	0.01	-	1.01	0.29	-	0.26	0.17	-
1.7	1.82	0.01	-	0.73	0.12	-	0.28	0.17	-
1.8	2.56	0.01	-	0.80	0.06	-	0.26	0.26	-
1.9	1.85	0.01	-	0.92	0.06	-	0.25	0.16	-
2.0	2.38	0.01	1.51	1.20	0.09	0.20	0.27	0.17	0.39
2.1	2.81	0.02	9.98	1.23	0.06	0.49	0.30	0.18	0.47
2.2	2.29	0.02	8.55	1.74	0.04	0.53	0.29	0.15	0.40
2.3	2.95	0.02	13.49	1.94	0.03	1.61	0.32	0.16	0.42
2.4	2.93	0.02	12.81	2.25	0.03	0.83	0.37	0.16	0.45
2.5	3.37	0.02	6.70	3.01	0.02	0.60	0.28	0.15	0.40
2.6	3.96	0.02	14.11	3.88	0.02	0.78	0.32	0.18	0.38
2.7	1.80	0.01	14.49	3.15	0.02	0.66	0.27	0.16	0.38
2.8	1.58	0.01	11.95	3.75	0.02	0.70	0.29	0.14	0.36
2.9	1.19	0.01	11.17	4.68	0.01	0.60	0.26	0.16	0.34
3.0	1.05	0.01	7.76	4.78	0.02	0.51	0.26	0.18	0.33
3.1	0.86	0.01	8.79	4.29	0.02	0.78	0.29	0.18	0.32
3.2	0.49	0.01	6.05	4.41	0.01	0.94	0.32	0.17	0.31
3.3	0.32	0.01	10.87	7.80	0.02	0.72	0.31	0.16	0.31
3.4	0.17	0.01	6.48	5.89	0.01	0.66	0.28	0.15	0.29
3.5	1.03	0.01	2.50	3.51	0.02	0.20	0.25	0.15	0.25
3.6	0.06	0.01	0.73	2.92	0.02	0.09	0.24	0.16	0.24
3.7	0.04	0.01	0.84	1.70	0.02	0.11	0.24	0.15	0.22
3.8-4.0	0.01	0.01	0.01	0.13	0.02	0.06	0.23	0.15	0.22

Table 7: Solution time (seconds) when $N = 10$, $M = 4$ for all DVFS schemes.

	TL-DVFS			PL-DVFS			SL-DVFS		
D	EFTMA	RAM	TDM	EFTMA	RAM	TDM	EFTMA	RAM	TDM
0.5	-	-	-	-	-	-	-	-	-
0.6	6.79	6.84	-	57.62	9.78	-	0.51	0.21	-
0.7	10.72	2.82	-	45.81	8.93	-	0.55	0.25	-
0.8	10.03	0.45	-	76.49	6.39	-	0.43	0.25	-
0.9	37.59	0.02	-	110.45	2.03	-	0.49	0.23	-
1.0	37.44	0.01	0.26	112.50	0.51	2312.10	0.34	0.20	0.94
1.1	60.24	0.01	3755.76	183.62	0.30	110698.16	0.46	0.20	1.17
1.2	830.27	0.01	2283.87	269.00	0.50	4260.27	0.48	0.20	0.91
1.3	217.16	0.01	1335.57	566.26	0.08	38399.02	0.58	0.21	1.05
1.4	215.56	0.01	702.14	424.56	0.03	46466.88	0.67	0.21	0.85
1.5	273.33	0.01	2991.75	438.05	0.03	196016.53	0.56	0.20	0.91
1.6	9.52	0.01	3390.81	191.38	0.03	162484.17	0.48	0.20	0.76
1.7	1.16	0.01	214.75	35026.12	0.03	120531.33	0.50	0.21	0.48
1.8	5.62	0.01	13.83	31.67	0.03	41.78	0.32	0.20	0.30
1.9-3.0	0.03	0.02	0.03	0.58	0.04	0.33	0.29	0.22	0.27

References

1. R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

Table 8: Solution time (seconds) when $N = 20$, $M = 2$ for all DVFS schemes.

	TL-DVFS			PL-DVFS			SL-DVFS		
D	EFTMA	RAM	TDM	EFTMA	RAM	TDM	EFTMA	RAM	TDM
2.2	-	-	-	-	-	-	-	-	-
2.4	3.38	9.17	-	3.46	0.27	-	NaN	0.09	-
2.6	2.95	26.80	-	6.39	0.80	-	0.66	0.20	-
2.8	2.47	47.22	-	3.10	0.12	-	0.45	0.16	-
3.0	1.42	0.23	-	2.07	0.10	-	0.20	0.17	-
3.2	2.25	1.07	-	2.44	0.10	-	0.11	0.16	-
3.4	2.36	0.01	-	3.78	0.09	-	0.16	0.18	-
3.6	1.59	0.01	-	3.65	0.11	-	0.28	0.17	-
3.8	2.55	0.01	-	4.79	0.27	-	0.11	0.18	-
4.0	2.39	0.01	-	5.50	0.21	-	0.13	0.19	-
4.2	2.43	0.01	-	4.73	0.46	-	0.27	0.16	-
4.4	3.31	0.01	-	4.26	0.13	-	0.34	0.22	-
4.6	4.62	0.01	6.26	4.25	0.81	0.56	0.25	0.15	0.36
4.8	4.87	0.01	7.04	3.45	0.48	0.67	0.23	0.17	0.40
5.0	4.28	0.01	2.44	3.46	0.08	0.79	0.25	0.17	0.41
5.2	4.51	0.01	4.28	7.76	0.05	1.33	0.22	0.20	0.47
5.4	3.85	0.01	4.36	28.74	0.04	1.18	0.23	0.17	0.48
5.6	2.97	0.01	5.21	48.56	0.03	1.50	0.30	0.17	0.47
5.8	2.67	0.01	7.40	73.50	0.03	0.96	0.59	0.17	0.45
6.0	1.51	0.01	6.33	98.67	0.06	0.97	0.20	0.22	0.34
6.2	2.13	0.02	2.94	65.51	0.06	0.47	0.23	0.17	0.34
6.4	0.82	0.02	2.00	42.77	0.04	0.40	0.19	0.15	0.27
6.6	0.24	0.02	0.63	6.06	0.03	0.34	0.31	0.14	0.29
6.8	0.03	0.01	1.13	8.28	0.03	0.79	0.19	0.17	0.31
7.0	0.05	0.01	0.19	11.32	0.03	0.31	0.20	0.21	0.28
7.2	0.02	0.01	0.42	13.92	0.03	0.28	0.36	0.18	0.27
7.4	0.02	0.01	0.02	0.38	0.03	0.27	0.17	0.18	0.29
7.6	0.02	0.01	0.02	0.39	0.03	0.27	0.55	0.27	0.27

Table 9: Solution time (seconds) when $N = 20$, $M = 4$ for all DVFS schemes.

	TL-DVFS			PL-DVFS			SL-DVFS		
D	EFTMA	RAM	TDM	EFTMA	RAM	TDM	EFTMA	RAM	TDM
1.0	-	-	-	-	-	-	-	-	-
1.2	2576.81	63381.17	-	207.57	99.97	-	0.66	0.30	-
1.4	1909.92	57079.83	-	7040.64	172.35	-	0.58	0.17	-
1.6	521.33	791.30	-	1401.37	25.66	-	0.40	0.15	-
1.8	673.87	0.08	-	16611.93	7.37	-	0.26	0.16	-
2.0	399.08	0.07	-	8037.54	0.98	-	0.35	0.18	-
2.2	1237.58	0.04	-	24671.73	0.12	-	0.41	0.17	-
2.4	4356.88	0.05	189.97	2041.74	0.13	17625.22	0.50	0.17	1.38
2.6	165.41	0.05	1193.19	48888.64	0.15	19109.32	0.83	0.24	2.12
2.8	38.65	0.05	1447.58	76090.41	0.12	82873.64	5.30	0.18	3.71
3.0	25.84	0.04	4258.07	86462.91	0.11	8431.17	1.02	0.18	0.88
3.2	3.80	0.05	31173.63	47538.99	0.12	11296.63	0.78	0.23	0.45
3.4	0.19	0.04	4917.95	202.77	0.17	3509.51	0.63	0.33	0.43
3.6	0.87	0.05	11.45	34489.84	0.07	63.94	0.38	0.25	0.25
3.8-6.0	0.04	0.02	0.03	1.64	0.06	0.52	0.28	0.15	0.14

2. T. Sudo, H. Sasaki, and J. L. Drewniak, "Electromagnetic interference (emi) of system-on-package (sop)," *IEEE Trans. Advanced Packaging*, vol. 27, no. 2, pp. 304–314, 2004.

3. M. Mbrahimi, A. Evans, M. B. Tahoori *et al.*, "Comprehensive analysis of alpha and neutron particle-induced soft errors in an embedded processor at nanoscales," *IEEE/ACM Design, Automation & Test in Europe*, 2014.

4. D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," *IEEE/ACM Int. Conf. Computer Aided Design*, pp. 35–40, 2004.
5. B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. Industrial Informatics*, vol. 6, no. 3, pp. 316–328, 2010.
6. 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.
7. M. A. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Trans. Parallel and Distributed Systems*, vol. 28, no. 3, pp. 813–825, 2017.
8. J. W. McPherson, "Reliability challenges for 45nm and beyond," *ACM Design Automation Conference*, 2006.
9. B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," *ACM Trans. Design Automation of Electronic Systems*, vol. 18, no. 2, pp. 1–21, 2013.
10. G. Xie, Y. Chen, Y. Liu *et al.*, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems," *IEEE Trans. Industrial Informatics*, vol. 13, no. 4, pp. 1629–1640, 2017.
11. Z. Deng, D. Cao, H. Shen *et al.*, "Reliability-aware task scheduling for energy efficiency on heterogeneous multiprocessor systems," *The Journal of Super computing*, p. 11643–11681, 2021.
12. G. Xie, Y. Chen, X. Xiao *et al.*, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Sustainable Computing*, vol. 3, no. 3, pp. 167–181, 2018.
13. J. Zhou, J. Sun, X. Zhou *et al.*, "Resource management for improving soft-error and lifetime reliability of real-time MPSoCs," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2215–2228, 2019.
14. M. Cui, L. Mo, A. Kritikakou, and E. Casseau, "Embedded computer systems: Architectures, modeling, and simulation(samos), springer," *Embedded Computer Systems: Architectures, Modeling, and Simulation(SAMOS), Springer*, pp. 213–227, 2020.
15. L. Han, L. C. Canon, J. Liu, Y. Robert *et al.*, "Improved energy-aware strategies for periodic real-time tasks under reliability constraints," *2019 IEEE Real-Time Systems Symposium (RTSS)*, pp. 17–29, 2019.
16. 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.
17. S. K. Roy, R. Devaraj, A. Sarkar, K. Maji *et al.*, "Contention-aware optimal scheduling of real-time precedence-constrained task graphs on heterogeneous distributed systems," *Journal of Systems Architecture*, vol. 105, 2020.
18. Z. Quan, Z. J. Wang, T. Ye *et al.*, "Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 31, pp. 1165 – 1182, 2020.
19. K. Cao, J. Zhou, P. Cong *et al.*, "Affinity-driven modeling and scheduling for makespan optimization in heterogeneous multiprocessor systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, pp. 1189 – 1202, 2018.
20. H. Xu, R. Li, C. Pan *et al.*, "Minimizing energy consumption with reliability goal on heterogeneous embedded systems," *Journal of Parallel and distributed Computing*, vol. 127, pp. 44 – 57, 2019.
21. L. Zhang, K. Li, K. Li *et al.*, "Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems," *Int. Journal of Electrical Power Energy Systems*, vol. 78, pp. 499–512, 2016.
22. Y. Guo, D. Zhu, and H. Aydin, "Reliability-aware power management for parallel real-time applications with precedence constraints," in *IEEE Int. Green Computing Conf.*, 2011, pp. 1–8.
23. 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.

24. 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 International Symposium on Low Power Electronics and Design (ISLPED)*, 2015, pp. 225–230.
25. M. Salehi, A. Ejlali, and B. M. Al-Hashimi, "Two-phase low-energy n-modular redundancy for hard real-time multi-core systems," *IEEE Trans. Distributed Systems*, vol. 27, no. 5, pp. 1497–1510, 2016.
26. C. Gou, A. Benoit, M. Chen *et al.*, "Reliability-aware energy optimization for throughput-constrained applications on MPSoC," in *IEEE Int. Conf. Parallel and Distributed Systems*.
27. S. Tosun, "Energy- and reliability-aware task scheduling onto heterogeneous MPSoC architectures," *The Journal of Supercomputing*, vol. 62, 2012.
28. S. Moulik, R. Chaudhary, and Z. Das, "Hears: A heterogeneous energy-aware real-time scheduler," *Microprocessors and Microsystems*, vol. 72, 2020.
29. G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embedded Computing Systems*, vol. 13, no. 3, 2014.
30. K. Skadron, M. R. Stan, K. Sankaranarayanan *et al.*, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Tran. Architecture and Code Optimization*, vol. 1, no. 1, p. 94–125, Mar. 2004.
31. 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 Int. Conf. on Computer-Aided Design*, 2019.
32. G. Quan and V. Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *IEEE Trans. Industrial Informatics*, vol. 6, no. 3, pp. 329–339, 2010.
33. 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.
34. J. Deverge and I. Puaut, "Safe measurement-based wcet estimation," in *WCET*, 2007.