Energy-aware Partial-Duplication Task Mapping under Real-Time and Reliability Constraints

Minyu Cui, Lei Mo, Angeliki Kritikakou, Emmanuel Casseau

Energy-aware Partial-Duplication Task Mapping under Real-Time and Reliability Constraints

Minyu Cui[1], Lei Mo[2], Angeliki Kritikakou[1], and Emmanuel Casseau[1]

[1] Univ Rennes, Inria, IRISA, CNRS, France
[2] School of Automation, Southeast University, China

Abstract. An efficient task execution on multicore platforms can lead to low energy consumption. To achieve that, an Integer Non-Linear Programming (INLP) formulation is proposed that performs task mapping by jointly addressing task allocation, task frequency assignment, and task duplication. The goal is to minimize energy consumption under real-time and reliability constraints. To provide an optimal solution, the original INLP problem is safely transformed to an equivalent Mixed Integer Linear Programming (MILP) problem. The comparison of the proposed approach with existing energy-aware task mapping approaches shows that the proposed approach is able to find solutions when other approaches fail, achieving an overall lower energy consumption.

Keywords: Reliability · Task mapping · DVFS · multicores · Real-time.

1 Introduction

The execution of embedded applications on multicore platforms requires both in-time and reliable execution of tasks. In time-critical domains, tasks must finish before a given deadline. Meanwhile, their correct execution can be threatened by several sources, such as radiation [6] and electromagnetic interference [7], causing non-persistent *transient faults* (also called *soft errors*). Due to the technology size reduction and the increasing scaling of CMOS technology, the system has become more susceptible to soft errors [5]. To provide protection, the system reliability should be increased. To achieve that, tasks are executed with higher frequencies [8]. Another way is to apply fault-tolerance techniques at the system level, where tasks are replicated [2,4]. Nonetheless, these approaches increase the system energy consumption, which has become an important concern, especially for systems with limited energy budget, such as battery-powered or energy-harvesting devices [2–4], and green computing. To balance system performance and energy consumption, multicore platforms have been enhanced with Dynamic Voltage and Frequency Scaling (DVFS). However, by decreasing frequency and voltage, systems become more susceptible to soft errors [9]. Therefore, adequate *mapping approaches* are required in order to obtain a reliable, in-time and energy efficient task execution on multicore platforms.

Table 1 depicts representative task mapping approaches, which decide the frequency assignment and the processor allocation to tasks, under Energy Budget

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

Ref.	Objective		Task replicas		Platform		DVFS	Constraints		
	min E	max R	Full	Partial	Homo.	Hete.		EB	RT	R
[2]		√			√		√	√		√
[4]		√				√		√	√	
[3]		√	√		√		√	√	√	
[15]		√	√			√	√			√
[10]	√					√	√		√	
[13]	√					√	√			√
[1]	√		√		√		√		√	√
[14]	√		√			√	√			√
[12]	√			√	√		√		√	
proposed	√			√	√		√		√	√

(EB), Real-Time (RT) and Reliability (R) constraints. The goal is to maximize system reliability ($\max R$), usually under an energy supply constraint [2,3] [4], or to minimize energy consumption ($\min E$), usually under a real-time [1, 10, 12] or reliability constraint [1, 13, 14]. Existing approaches execute only original tasks or, additionally, replicas of original tasks. A task is executed successfully, if at least one replica is executed without faults [1]. By executing several replicas on different processors, the probability of correct execution is increased, as it is unlikely that execution of all replicas fails [1, 14, 15]. Task replication can be either full (all tasks are replicated) or partial (selected tasks are replicated).

Task mapping approaches, that maximize reliability, consider i) only original tasks, through online heuristics [4] and static and dynamic algorithms [2], and ii) original and replicated tasks, when the number of replicas is known [3] and when the number of task replicas is also decided during the mapping process [15].

One category of the approaches, that minimize energy consumption, considers only original tasks, e.g., a decomposition algorithm is proposed under reliability constraints [13]. Mapping of original tasks on heterogeneous multicores is extended in order to provide a solution, where enough slack exist to re-execute a task in case of a single fault [10]. However, these approaches usually require high frequencies to satisfy the reliability constraints, increasing energy consumption. On top, even with the highest platform frequency, it may be impossible to satisfy the reliability constraints, leading to empty solution space. Another category considers both original and replicated tasks. Full replication approaches decide the number of replicas (at least one replica per task), e.g., for homogeneous [1] and heterogeneous [14] platforms. However, full replication 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. This impact may lead to empty solution space, when the real-time constraints are strict. Few approaches apply partial task replication. Usually heuristics are applied to select the tasks to be replicated, e.g., a heuristic that decides the tasks to be duplicated and their frequency [12], without considering reliability constraints. Our work belongs to this category and extends the State-of-the-Art (SoA) by optimally solving a combined task mapping problem, under both real-time and reliability constraints.

The goal of the proposed task mapping approach is to minimize system energy consumption by optimally and concurrently deciding the tasks to be duplicated, the frequency assignment per task and task allocation, on a homogeneous multicore platform with DVFS, under both real-time and reliability constraints. The proposed approach considers the execution of a task to be safe, when it meets its reliability constraint. Otherwise the task is duplicated. As task duplication is applied only when the reliability constraint cannot be met, the number of duplicated tasks is reduced, reducing the negative impact on time and energy consumption. Our contribution is two-fold: i) the aforementioned task mapping problem is formulated as an Integer Non-Linear Programming (INLP), and ii) the INLP problem is safely transformed into a Mixed Integer Linear Programming (MILP) problem in order to be solved optimally. To support our contribution, we perform extensive experiments considering i) strict, medium and relaxed real-time constraints, and ii) strict and relaxed reliability constraints. From the obtained results, our approach is able to find more feasible solutions (on average, 31.77%) compared to approaches that always satisfy the reliability constraints without task replication, and consumes less energy (on average, 26.07%) compared to full duplication approaches. Especially for strict real-time constraints, our approach can achieve energy savings up to 51.56%.

The rest of the paper is organized as follows. Section 2 introduces the system model. Section 3 presents the proposed formulation of the duplication-based fault-tolerant mapping approach. Section 4 presents the evaluation results. Finally, Section 5 concludes this study.

2 System Model

Task Model: We consider a set of N independent tasks, i.e., $\{\tau_1, \ldots, \tau_N\}$. For each task τ_i, C_i is the Worst Case Execution Cycles (WCEC). All tasks must be executed before a common deadline D, and no preemption occurs between different tasks executed on the same processor. Without loss of generality, in the rest of the paper, we assume that the release times of all tasks are at the start of the scheduling period.

Platform and Power Model: A multicore platform is considered with M homogeneous processors, i.e., $\{\theta_1, \ldots, \theta_M\}$. The multicore platform supports individual DVFS in task level and there are L different voltage/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 given by $t_i = \frac{C_i}{f_l}$. For each processor θ_m, the power consumption is modeled as the sum of static P_l^{sta} power and dynamic P_l^{dyn} power with frequency f_l [11], i.e., $P_l = P_l^{sta} + P_l^{dyn}$. The dynamic power consumption with voltage/frequency 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 an average fault rate $\lambda(f)$ at frequency f, which can be modeled as $\lambda(f) = \lambda_0 \times 10^{d \frac{f_{\max} - f}{f_{\max} - f_{\min}}}$, 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 [8]. The reliability of a task τ_i, i.e., the probability of executing the task without any fault, follows the exponential model [8]. Hence, the task's reliability at frequency f_l is $r_i(f_l) = e^{-\varphi_i(f_l)}$, where $\varphi_i(f_l) = \lambda(f_l) \times t_i$, with t_i the execution time of task τ_i at frequency f_l [2, 8, 10]. If the reliability of task τ_i is larger than its reliability constraint, the execution is considered as reliable and the task reliability is not modified, $R_i = r_i(f_l)$. Otherwise, the task τ_i is duplicated and executed on different processors. The duplicated task is executed with same frequency as the original task, similar to [1, 12, 16], since using the same frequency in original and duplicated tasks yields similar energy consumption [16]. When the task is duplicated, its reliability is given by $R_i = 1 - (1 - r_i(f_l))^2$. As our approach is applied offline, 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 platform workload, in case no faults occur. An mechanism can be applied, during execution, to further improve the energy consumption of our solution (by not executing the replica, when the first execution is correct), such as [17], and to deal with the low probability cases, where non-replicated tasks and both original and duplicated tasks are faulty, such as [18].

3 Duplication-based Fault-Tolerant Mapping Approach (DFTMA)

Motivational example: Before providing the mathematical formulation of the proposed Duplication-based Fault-Tolerant Mapping Approach (DFTMA), we provide a simple example to motivate our approach. Let us assume a single task with $C{=}4 * 10^8$, a reliability threshold equal to 0.999 and the voltage/frequency levels of the platform from Table 4 in section 4. Table 2 depicts the execution time (t), energy consumption ($E = P_l \times t$) and reliability R, when only the original task is executed (columns $f_1 - f_6$) and when both original and its replica are executed (columns $2f_1 - 2f_6$). Table 2.a) enumerates all feasible solutions under only the reliability constraint for the approaches that i) apply full task

Table 2: Motivational Example

$\boldsymbol{f_l}$	f_1	f_2	f_3	f_4	f_5	f_6	$2f_1$	$2f_2$	$2f_3$	$2f_4$	$2f_5$	$2f_6$
$\boldsymbol{t}$	0.4994	0.4825	0.4677	0.4547	0.4431	0.4	0.9988	0.965	0.9444	0.9094	0.8862	0.8
$\boldsymbol{E}$	2.1169	2.7905	3.6959	4.926	6.6141	8.9525	4.2338	5.581	7.3918	9.852	13.2282	17.905
$\boldsymbol{R}$	0.9753	0.9909	0.9965	0.9985	0.9994	1	0.99939	0.99992	0.99999	0.99999	0.999999	1

a) Feasible solutions of three approaches under reliability constraint

DFTMA	$A_1{=}f_6$, $A_2{=}f_5$, $A_3{=}2f_4$, $A_4{=}2f_3$, $A_5{=}2f_2$, $A_6{=}2f_1$
FTM	$B_1{=}f_6$, $B_2{=}f_5$
FDM	$C_1{=}2f_6$, $C_2{=}2f_5$, $C_3{=}2f_4$, $C_4{=}2f_3$, $C_5{=}2f_2$, $C_6{=}2f_1$

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

$\boldsymbol{\leq D}$	0	0.4	0.4431	0.8	0.8862	0.9094	0.9444	0.965	0.9988
$\boldsymbol{D <}$	0.4	0.4431	0.8	0.8862	0.9094	0.9444	0.965	0.9988	-
DFTMA	-	$\boldsymbol{A_1}$	A_1,$\boldsymbol{A_2}$	A_1,$\boldsymbol{A_2}$	A_1,$\boldsymbol{A_2}$	A_1,$\boldsymbol{A_2}$,A_3	A_1,$\boldsymbol{A_2}$,A_3,A_4	A_1-A_4,$\boldsymbol{A_5}$	A_1-A_5,$\boldsymbol{A_6}$
FTM	-	$\boldsymbol{B_1}$	B_1, $\boldsymbol{B_2}$	B_1, $\boldsymbol{B_2}$	B_1, $\boldsymbol{B_2}$	B_1, $\boldsymbol{B_2}$	B_1, $\boldsymbol{B_2}$	B_1, $\boldsymbol{B_2}$	B_1, $\boldsymbol{B_2}$
FDM	-	-	-	$\boldsymbol{C_1}$	C_1, $\boldsymbol{C_2}$	C_1, C_2, $\boldsymbol{C_3}$	C_1-C_3, $\boldsymbol{C_4}$	C_1-C_4, $\boldsymbol{C_5}$	C_1-C_5, $\boldsymbol{C_6}$

duplication (FDM), such as [13], ii) must always satisfy the reliability constraint without duplication (FTM), such as [1, 14], and iii) apply task duplication only when the reliability constraint cannot be satisfied (DFTMA), such as the proposed method. Table 2.b) explores how the deadline constraint (D) affects the feasible and optimal (highlighted in bold) solutions. Compared to DFTMA approach, existing approaches cannot find optimal solutions, due to strict reliability constraint (as FTM) or deadline constraint (as FDM), even for only a single task.
Mathematical formulation: 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 the: 1) frequency of original tasks ($\boldsymbol{s}$); 2) duplication of original tasks ($\boldsymbol{\sigma}$); 3) allocation of original tasks ($\boldsymbol{q}$) and duplicated tasks ($\boldsymbol{d}$). Table 3 summarizes the parameters and the variables of our formulation.

i) Frequency Assignment: Let $\mathbf{N} = \{1, \ldots, N\}$, $\mathbf{M} = \{1, \ldots, M\}$ and $\mathbf{L} = \{1, \ldots, L\}$. Since the platform provides individual DVFS, that can be applied per task, each task can only be assigned with one frequency level:

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

ii) Task Duplication Decision: The reliability of 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 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 $x > y$, $c = 0$, then $\delta - (1 + \delta)c \leq x - y \leq 1 - c$, 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, then $c = 1$. 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 $x > y$,

Parameters	
M	number of processors
N	number of tasks
L	number of discrete V/F levels
C_i	WCEC of task τ_i
D	global deadline
(v_l, f_l)	the l^{th} V/F level
P_l	the l^{th} power level of processor
R_i^{th}	task τ_i reliability constraint

Binary Variables	
s_{il}	1 if task τ_i executes with frequency f_l 0 otherwise
q_{im}	1 if task τ_i executes on processor θ_m 0 otherwise
σ_i	1 if task τ_i is duplicated 0 otherwise
d_{im}	1 if duplication of τ_i executes on θ_m 0 otherwise

Table 3: Main Notations

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, then $r_i \in \{e^{-\varphi_i(f_1)}, \ldots, e^{-\varphi_i(f_L)}\}$. According to Lemma 1, the relationship between $R_i(f_l)$, R_i^{th} and σ_i is linearized as follows:

$$\delta - (1+\delta)\sigma_i \leq \sum\nolimits_{l \in \boldsymbol{L}} s_{il} e^{-\varphi_i(f_l)} - R_i^{th} \leq 1 - \sigma_i, \ \forall i \in \boldsymbol{N}. \tag{2}$$

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

$$\sum\nolimits_{m \in \boldsymbol{M}} q_{im} = 1, \ \forall i \in \boldsymbol{N}. \tag{3}$$

iv) Duplicated task allocation: If a task τ_i is duplicated, the original and duplicated tasks should be allocated on different processors. The duplicated task is allocated only on one processor:

$$\sum\nolimits_{m \in \boldsymbol{M}} d_{im} = \sigma_i, \ \forall i \in \boldsymbol{N}, \tag{4a}$$

$$q_{im} + d_{im} \leq 1, \ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}. \tag{4b}$$

v) Real-time Constraints: All (original and duplicated) tasks assigned on processor θ_m should be executed within a common deadline D:

$$\sum\nolimits_{i \in \boldsymbol{N}} q_{im} t_i + \sum\nolimits_{i \in \boldsymbol{N}} d_{im} t_i \leq D, \ \forall m \in \boldsymbol{M}, \tag{5}$$

where $t_i = \sum_{l \in \boldsymbol{L}} s_{il} \frac{C_i}{f_l}$ is the execution time of task τ_i.

vi) Objective Function: As the duplicated task is executed with the frequency of the original task, their execution consumes the same energy. Therefore, the energy consumption of original task τ_i and its potential duplicated task is:

$$E_i = \sum\nolimits_{l \in \boldsymbol{L}} (1 + \sigma_i) s_{il} P_l \frac{C_i}{f_l}. \tag{6}$$

Based on Equation (6), the total energy consumed by the system until the deadline is $E_s = \sum_{i \in \boldsymbol{N}} E_i$. Based on the objective function and the aforementioned constraints, the Primal Problem (**PP**) is formulated as

$$\mathbf{PP}: \min_{\boldsymbol{s},\boldsymbol{q},\boldsymbol{\sigma},\boldsymbol{d}} E_s \tag{7}$$

$$\text{s.t.} \begin{cases} (1), (2), (3), (4a), (4b), (5), \\ s_{il}, q_{im}, \sigma_i, d_{im} \in \{0, 1\}, \ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}, \ \forall l \in \boldsymbol{L}. \end{cases}$$

Since Equation (5) and Equation (6) include the nonlinear items $q_{im}t_i$, $d_{im}t_i$ and $\sigma_i s_{il}$, **PP** is an INLP problem, which is difficult to solve optimally. In order to find the optimal solution, and simplify the structure of problem, we equivalently transform **PP** to an MILP problem. By applying *variable replacement*, the nonlinear variable combinations are replaced and an equal to the original MILP formulation is obtained. To this end, we introduce 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. The inequalities $g \leq b_1$ and $g \leq b_2$ ensure that g will be zero, if either b_1 or b_2 is zero. The equality $g \geq b_1 + b_2 - 1$ ensures that g becomes 1, if both variables b_1 and b_2 are equal to 1.

According to Lemma 2, we introduce the auxiliary binary variable h_{il} and let $h_{il} = \sigma_i s_{il}$. Then, Equation (6) can be replaced by:

$$E_i = \sum\nolimits_{l \in \boldsymbol{L}} \left(s_{il} P_l \frac{C_i}{f_l} + h_{il} P_l \frac{C_i}{f_l} \right), \ \forall i \in \boldsymbol{N}, \tag{8a}$$

$$-s_{il} + h_{il} \leq 0, -\sigma_i + h_{il} \leq 0, s_{il} + \sigma_i - h_{il} \leq 1, \ \forall i \in \boldsymbol{N}, \ \forall l \in \boldsymbol{L}. \tag{8b}$$

As a next step, $q_{im} t_i$ and $d_{im} t_i$ are linearized. Since $s_{il} \in \{0, 1\}$ and C_i is large enough, one cycle has a negligible impact on the solution. Thus, t_i are relaxed to continuous variables: $0 \leq t_i = \sum_{l \in \boldsymbol{L}} s_{il} \frac{C_i}{f_l} \leq \overline{T_i}$, where $\overline{T_i} = \frac{C_i}{f_{min}}$. The following lemma is introduced:

Lemma 3. *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 3, the continuous variables $\chi_{im} = q_{im} t_i$ and $\xi_{im} = d_{im} t_i$ are introduced, replacing Equation (5) by:

$$\sum\nolimits_{i \in \boldsymbol{N}} \chi_{im} + \sum\nolimits_{i \in \boldsymbol{N}} \xi_{im} \leq D, \ \forall m \in \boldsymbol{M}, \tag{9a}$$

$$-\overline{T_i} q_{im} + \chi_{im} \leq 0, \ -t_i + \chi_{im} \leq 0, \ \overline{T_i} q_{im} + t_i - \chi_{im} \leq \overline{T_i}, \ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}, \tag{9b}$$

$$-\overline{T_i} d_{im} + \xi_{im} \leq 0, \ -t_i + \xi_{im} \leq 0, \ \overline{T_i} d_{im} + t_i - \xi_{im} \leq \overline{T_i}, \ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}. \tag{9c}$$

Therefore, the primal problem (7) is equally reformulated as follows:

$$\mathbf{PP1}: \min_{\substack{\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{\sigma}, \boldsymbol{d}, \boldsymbol{h}, \\ \boldsymbol{t}, \boldsymbol{\chi}, \boldsymbol{\xi}}} \sum\nolimits_{i \in \boldsymbol{N}} \sum\nolimits_{l \in \boldsymbol{L}} P_l \frac{C_i}{f_l} (s_{il} + h_{il}) \tag{10}$$

$$\text{s.t.} \begin{cases} (1) \sim (3), (4a), (4b), (8b), (9a) \sim (9c), \\ -\sum_{l \in \boldsymbol{L}} \frac{C_i}{f_l} s_{il} + t_i = 0, \ \forall i \in \boldsymbol{N}, \\ s_{il}, q_{im}, \sigma_i, d_{im}, h_{il} \in \{0, 1\}. \\ 0 \leq t_i, \chi_{im}, \xi_{im} \leq \overline{T_i}, \ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}, \ \forall l \in \boldsymbol{L}. \end{cases}$$

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

4 Experimental Results

Experimental set-up: The processor model 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 [20]. To obtain realistic bounds for the WCEC, we analyse typical benchmarks, such as `stringsearch`, `qsort`, `blowfish`, and `dikstra` (MiBench suite [21]) and integer matrix multiplication (`matmul`). Through a measurement-based approach using Comet simulator, we count the number of execution cycles and memory access (Table 4). To obtaining safe and context-independent measurement, the sources of timing variability are eliminated [22], i.e., data-caches are considered disabled, the task is executed without interferences on a single core (WCEC_{iso}). Taking into account the number of cores, the number of memory accesses, and the delay of accessing the main memory, we compute WCEC_{inf} considering interferences. Regarding the DVFS, $L = 6$ voltage/frequency levels are used, based on the work of [19] considering 64 nm technology, as depicted Table 4.

To explore the ability to find optimal solutions and the solution quality, we performed a large and diverse set of experiments. More precisely, we tune the i) number of tasks, ii) WCEC, iii) reliability constraints and iv) deadline, considering 4 processors ($M = 4$). The failure rate constant has been set to $d_0 = 3$ and the initial failure rate $\lambda_0 = 5 \times 10^{-6}$ faults per second [8]. Three task sets are considered with N=10, 20 and 30 tasks. We generate the tasks WCEC randomly based on the measured WCEC values of Table 4, i.e., within the range $[1 \times 10^8, 4 \times 10^8]$, incorporating the cost of the DVFS switching mechanism. The deadline is given by $D = k \times \frac{N}{M} \times \frac{1}{2}(\frac{C_{max}}{f_{min}} + \frac{C_{max}}{f_{max}})$, where k controls the range of the deadline, from tight deadlines up to more relaxed ones. The reliability constraint is randomly selected using Low Reliability Threshold (LRT) range $[0.9990, 0.9995]$ or a High Reliability Threshold (HRT) range $[0.9995, 1]$ based on the magnitude 10^{-3} of reliability target in [1]. Notice that, these numbers

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	$\boldsymbol{WCEC_{iso}}$ (Cyc.)	**Num. Accesses**	$\boldsymbol{WCEC_{inf}}$ **(Cyc.)**
`matmul int 32x32`	3,313,958	371,957	226,488,158
`matmul int64 32x32`	4,055,289	507,133	308,335,089
`qsort int 1000`	875,616	184,089	111,329,016
`qsort int64 1000`	1,219,854	259,553	156,951,654
`qsort softfloat 1000`	1,745,122	185,437	113,007,322
`dijkstra`	766,369	117,151	71,056,969
`blowfish`	3,058,991	110,330	69,256,991
`stringsearch`	13,093,544	597,608	371,658,344

change only the values of the problem parameters, and not the problem structure. The evaluation is performed following the Once Tuning One Parameter (OTOP) approach, where only one parameter is modified at each experiment. Each experiment is repeated 20 times, modifying the WCEC and the reliability constraints. Each figure point presents the average result. In total, we have performed at least 12 (different deadline values) $\times$ 3 (number of tasks) $\times$ 20 (with different WCEC and reliability constraint) = 720 experiments, per approach.

The first set of experiments explores how the proposed approach behaves (energy consumption, percentage of duplicated tasks) with respect to i) real-time constraints and ii) reliability constraints. The second set of experiments compares the energy consumption, the feasibility (the percentage of the experiments where we obtained the optimal solution), the reliability improvement, and the estimation time, under reliability and real-time constraints for: i) the proposed Duplication-based Fault-Tolerant Mapping Approach (DFTMA), ii) a task mapping approach always satisfying the reliability constraint, without applying task duplication (formulated as **FTM**), similar to [13], and iii) a task mapping approach (formulated as **FDM**) always applying task duplication, similar to [1,14], when the number of replicas is set to two. We present results using LRT range, in order not to penalise the FTM approach. FTM under HRT constraint leads to many infeasible solutions. Since **FTM** and **FDM** are INLP problems, we apply the proposed transformation to convert them into an MILP problems, in order to be able to compare their optimal solutions. We have used Matlab 2018a to implement and solve (MILP solver) the problem formulation of DFTMA, FTM and FDM formulation. The simulations are performed on a quad-core 2.11 GHz Intel i7 processor and 16 GB RAM.

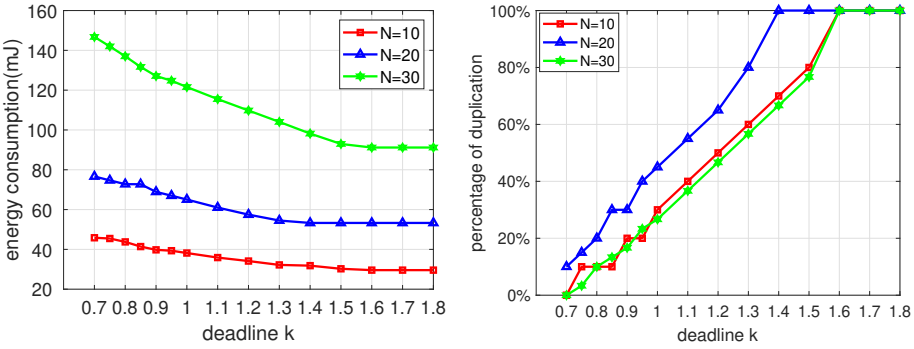
$$\textbf{FTM}: \min_{\boldsymbol{s},\boldsymbol{q}} \sum_{i\in\boldsymbol{N}} \sum_{l\in\boldsymbol{L}} s_{il} P_l \frac{C_i}{f_l}$$

$$\text{s.t.} \begin{cases} (1), (3), \\ \sum_{l\in\boldsymbol{L}} s_{il} e^{-\varphi_i(f_l)} \geq R_i^{th}, \ \forall i \in \boldsymbol{N}, \\ \sum_{i\in\boldsymbol{N}} q_{im} t_i \leq D, \ \forall m \in \boldsymbol{M}, \\ s_{il}, q_{im} \in \{0,1\}, \\ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}, \ \forall l \in \boldsymbol{L}. \end{cases}$$

$$\textbf{FDM}: \min_{\boldsymbol{s},\boldsymbol{q},\boldsymbol{d}} \sum_{i\in\boldsymbol{N}} \sum_{l\in\boldsymbol{L}} 2 s_{il} P_l \frac{C_i}{f_l}$$

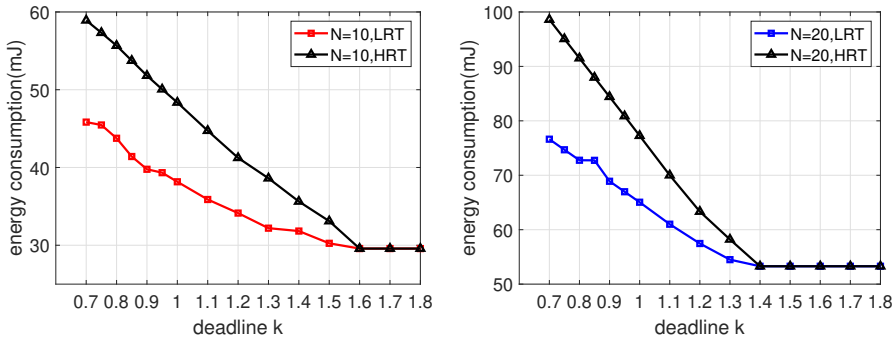
$$\text{s.t.} \begin{cases} (1), (3), (4b), (5), \\ \sum_{m=1}^{M} d_{im} = 1, \ \forall i \in \boldsymbol{N}, \\ s_{il}, q_{im}, d_{im} \in \{0,1\}, \\ \forall i \in \boldsymbol{N}, \ \forall m \in \boldsymbol{M}, \ \forall l \in \boldsymbol{L}. \end{cases}$$

Real-time constraint: Fig. 1a shows the energy consumption DFTMA, when the parameter k is tuned, from tight up to relaxed deadlines, with a reliability constraint within the LRT range. Fig. 1a shows that more energy is consumed: i) when the task number N increases, and ii) when real-time constraints become tight (k decreases), since higher frequencies are required to meet the deadline. With k increasing, more slack exists, and a lower frequency (that still meets the reliability constraints) can be assigned, minimizing energy consumption. From a point and on ($k = 1.6$ for $N = 10$, $k = 1.4$ for $N = 20$, $k = 1.5$ for $N = 30$), the energy consumption is stable; the deadline is relaxed enough, and thus, it does not influence the global optimal solution. Fig. 1b shows the percentage of duplicated tasks by DFTMA. The number of duplicated tasks is low at tight deadlines; no slack exists, thus duplication cannot be applied and the deadline met at the same time. As k increases, slack is created, tasks are duplicated and lower frequencies are assigned, reducing energy consumption. With large enough deadline, almost all tasks are duplicated and executed with low frequencies.



(a) Energy consumption (b) Duplication decision

Fig. 1: Energy consumption and duplication decision for LRT.



(a) $N = 10$ (b) $N = 20$

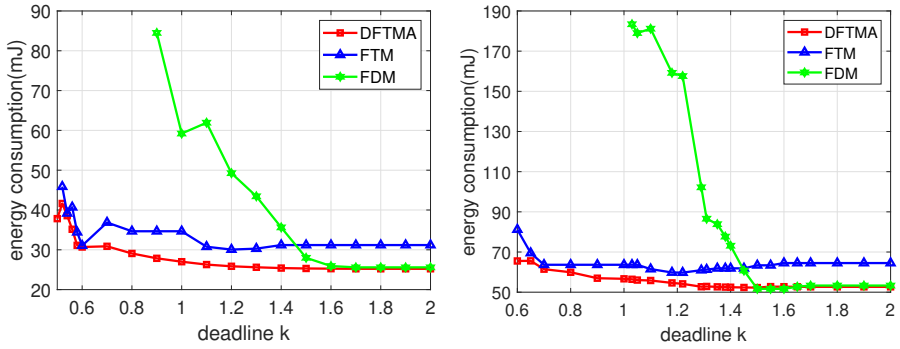
Fig. 2: Energy consumption considering LRT and HRT.

Reliability constraint: We further explore the behavior of DFTMA, when the reliability constraints belong to LRT and HRT regions. Fig. 2 shows the obtained energy consumption, when $N = 10$ (Fig. 2a) and $N = 20$ (Fig. 2b). The energy consumption decreases with more relaxed deadlines. The HRT case consumes more energy than LRT case under same conditions. As expect, when k is low, i.e., the deadline is strict, the strict reliability constraints require higher frequency to execute the tasks, which consumes more energy.

Energy consumption comparison: The energy consumption comparison of DFTMA, FTM and FDM approaches is depicted in Fig. 3a, with $N = 10$ and in Fig. 3b, with $N = 20$. A general observation is that the energy consumption obtained by DFTMA and FTM follows a similar trend. When the deadline is not strict, the energy consumption of DFTMA is less than the consumption of FTM. The FTM approach selects higher frequencies, since it requires to always satisfy the reliability constraints. On the contrary, DFTMA approach can exploit

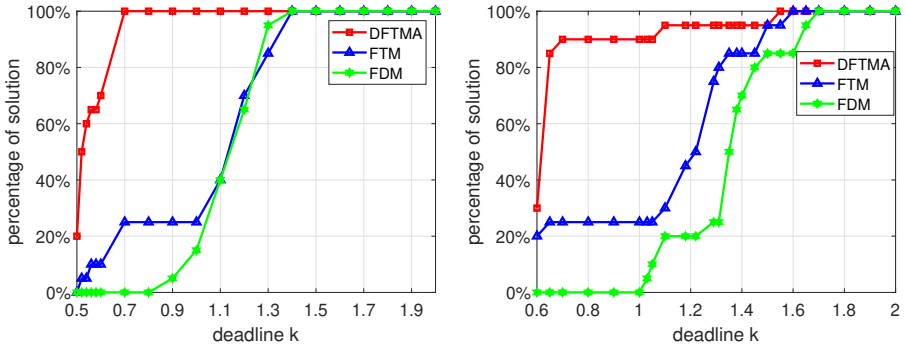
partial task duplication and use of lower frequencies in order to reduce energy consumption. The energy consumption of the FDM approach is much higher than both DFTMA and FTM approaches, especially when the real-time constraints are strict (e.g., k within the range $[0.9, 1.4]$ in Fig. 3a). High frequencies are required when deadline is strict in order to execute both original and duplicated tasks before the deadline, leading to high energy consumption. As k increases, lower frequencies are assigned, reducing energy consumption. With a relaxed enough deadline (after $k = 1.6$ in Fig. 3a and $k = 1.5$ in Fig. 3b), all tasks can be duplicated with low frequencies. DFTMA obtains the same solution in this case. DFTMA decides task duplication for all tasks after $k = 1.6$ with $N = 10$ and after $k = 1.4$ with $N = 20$, approaching the same behavior of FDM, which always applies duplication, achieving lower energy consumption than FTM.

Feasibility comparison: The percentage of solutions found by DFTMA, FTM and FDM approaches is depicted in Fig. 4a, when $N = 10$ and in Fig. 4b, when $N = 20$. A general observation is that the proposed DFTMA approach can find



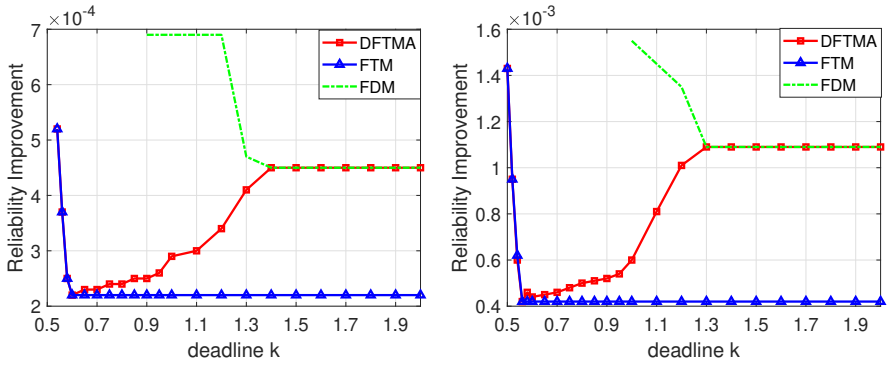
(a) $N = 10$ (b) $N = 20$

Fig. 3: Energy consumption comparison.



(a) $N = 10$ (b) $N = 20$

Fig. 4: Feasibility comparison



(a) $N = 10$ (b) $N = 20$

Fig. 5: Reliability improvement

the solution, even in cases where the FTM and FDM approaches are not able to find a solution, especially when the real-time constraint is strict, i.e., when k has a small value. The FDM approach has the worst behavior. In strict deadline cases (when k is small, $k = 0.5 \sim 0.9$ with $N = 10$ and $k = 0.6 \sim 1$ with $N = 20$), FDM cannot find any solution. This is explained since the FDM approach applies duplication for each task, thus more time is needed to execute the original and the duplicated tasks, compared to both the FTM (no duplication) and the proposed DFTMA (partial duplication of tasks). Compared with FTM, the proposed approach performs generally better: in FTM approach, the reliability is a hard constraint that must be always satisfied, whereas in DFTMA, if the reliability constraint cannot be satisfied by original task execution, then a duplication task is applied to achieve a reliable execution. In this way, the proposed approach has a higher probability of finding the solution. Both in Fig. 4a and Fig. 4b, we can observe that when deadline becomes relaxed enough, after a given point (for example when $k = 1.4$ in Fig. 4a and $k = 1.7$ in Fig. 4b), all the three approaches can almost 100% provide a solution. This is possible since the deadline constraints, after these points, become loose enough and do not impact the feasibility of obtaining a solution.

Reliability Improvement (RI): Fig. 5 shows the task reliability improvement above the reliability constraint, i.e., $RI = R_i - R_i^{th}$. The trend is the same for $N = 10$ and $N = 20$. For strict deadline, the RI is high for both DFTMA and FTM, due to the assignment of high frequencies, while FDM cannot find solutions. With deadline relaxing, FTM achieves a low RI, as it only requires to meet the reliability constraint. However, DFTMA exploits the time slack, duplicating tasks, increasing RI. Whenever FDM finds a solution, it has higher RI, due to full duplication. As k increases, DFTMA and FDM behave similarly.

Estimation time: Table 5 depicts the time required to obtain the solutions. For all approaches, the stricter is the deadline, the higher is the estimation time. Even if the proposed approach requires a few more seconds it is able to find a significantly higher number of solutions. For instance, when $N = 10$ and $k = 0.5$,

Table 5: Estimation time (seconds) when at least one solution is found

	N=10										N=20							
k	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7
DFTMA	8.36	0.14	0.13	0.05	0.16	0.19	0.06	0.06	0.09	0.06	114	1.23	0.16	0.13	0.13	0.2	0.05	0.06
FTM	3.1	0.06	0.11	0.03	0.13	0.06	0.02	0.03	0.03	0.02	10.75	0.56	0.11	0.08	0.06	0.14	0.02	0.02
FDM	-	-	-	-	179.03	23.81	0.05	0.03	0.02	0.03	-	94.7	20.66	0.09	0.03	0.02	0.09	0.11

DFTMA requires 5.26 sec more than FTM, but it finds solutions for 20% of the experiments, while the FTM finds very few ones. When $k = 0.7$, although the estimation time is similar, DFTMA is able to find optimal solutions for all experiments, while FTM found only 22% and FDM none. For the remaining cases, the estimation times are comparable. It should be stressed that as the approaches are applied offline, and thus, trading off a few more seconds in order to obtain optimal solutions is motivating.

5 Conclusion

In this paper, we studied the problem of task mapping by jointly solving task allocation, task frequency assignment, task duplication decision on multicore architectures enhanced with DVFS. The goal is to minimize energy consumption without violating the real-time constraint and taking into account the reliability. A safe transformation of the original INLP problem is applied to obtain an equivalent MILP problem, which is solved optimally. Through comparison of the proposed approach and two other energy-aware existing task mapping approaches, experimental results show that the proposed approach provides better trade-off between energy consumption and reliability, being able to find optimal solutions, even when existing approaches cannot. As future direction, the proposed strategy will be explored for heterogeneous architectures and for dependent task model.

Acknowledgement

This work has been founded by China Scholarship Council (CSC).

References

1. M. A. Haque, H. Aydin, and D. Zhu, "On Reliability Management of Energy-Aware Real-Time Systems Through Task Replication", *IEEE TPDS*, vol. 28, no. 3, pp. 813 - 825, 2017.
2. B. Zhao, H. Aydin, and D. Zhu,, "On Maximizing Reliability of Real-Time Embedded Applications Under Hard Energy Constraint", *IEEE TII*, vol. 6, no. 3, pp. 316-328, 2010.

3. J. Zhou, J. Sun, X. Zhou, T. Wei, M. Chen, S. Hu, and X. S. Hu, "Resource Management for Improving Soft-Error and Lifetime Reliability of Real-Time MPSoCs", *IEEE TCAD*, vol. 38, no. 12, pp. 2215-2228, 2019.
4. Y. Ma, T. Chantem, R. P. Dick, S. Wang, and X. S. Hu, "An On-Line Framework for Improving Reliability of real-Time Systems on 'Big-little' Type MPSoC", *IEEE/ACM DATE*, pp. 1558-1101, 2017.
5. M. Mbrahimi, A. Evans, M. B. Tahoori, R. Seyyedi, E. Costenaro, and D. Alexandrescu, "Comprehensive Analysis of Alpha and Neutron Particle-induced Soft errors in an embedded Processor at Nanoscales", *IEEE/ACM DATE*, pp. 1-6, 2014.
6. R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies", *IEEE TDMR*, vol. 5, no. 3, pp. 305-316, 2005.
7. T. Sudo, H. Sasaki, N. Masuda, and J. L. Drewniak, "Electromagnetic Interference (EMI) of System-on-Package (SOP)", *IEEE TAP*, vol. 27, no. 2, pp. 304-314, 2004.
8. D. Zhu, R. Melhem, and D. Mosse, "The Effects of Energy Management on Reliability in Real-Time Embedded Systems", *IEEE/ACM ICCAD*, pp. 35-40, 2004.
9. J. W. McPherson, "Reliability challenges for 45nm and beyond", *ACM DAC*, pp. 176-18, 2006.
10. K. Huang, X. Jiang, X. Zhang, R. Yan, K. Wang D. Xiong, and X. Yan, "Energy-efficient Fault-tolerant Mapping and Scheduling on Heterogeneous Multiprocessors Real-Time Systems", *IEEE Access*, vol. 6, pp. 57614-57630, 2018.
11. G. Chen, K. Huang, and A. Knoll, "Energy Optimization for Real-Time Multiprocessor System-on-Chip with Optimal DVFS and DPM Combination", *ACM TECS*, vol. 13, no. 3, 2014.
12. C. Gou, A. Benoit, M. Chen, L. Marchal, and T. Wei, "Reliability-aware energy optimization for throughput-constrained applications on MPSoC", *ICPADS*, 2018.
13. G. Xie, Y. Chen, Y. Liu, Y. Wei, R. Li, and K. Li, "Resource Consumption Cost Minimization of Reliable Parallel Applications on Heterogeneous Embedded Systems", *IEEE TII*, vol. 3, no. 3, pp. 1629 - 1640, 2017.
14. G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, and K. Li, "Energy-Efficient Fault-Tolerant Scheduling of Reliable Parallel Applications on Heterogeneous Distributed Embedded Systems", *IEEE TSC*, vol. 3, no. 3, pp. 167-181, 2018.
15. S. Wang, K. Li, J. Mei, G. Xiao, and K. Li, "A Reliability-aware Task Scheduling Algorithm Based on Heterogeneous Computing Systems", *JGC*, vol. 15, no. 1, pp. 23–39, 2017.
16. M. A. Haque, H. Aydin, and D. Zhu, "Energy-Aware Task Replication to manage reliability for periodic real-time applications on multicore platforms", *IEEE Int.Green Comput.Conf*, pp. 1-11, 2013.
17. Q. Zheng, B. Veeravalli, and C. Tham, "On the Design of Fault-Tolerant Scheduling Strategies Using Primary-Backup Approach for Computational Grids with Low Replication Costs," *IEEE TC*, vol. 58, no. 3, pp. 380-393, 2009.
18. E. Dubrova, *Fault Tolerant Design*. Springer, 2013.
19. G. Quan, and V. Chaturvedi, "Feasibility Analysis for Temperature-Constraint Hard Real-Time Periodic Tasks", *IEEE TII*, vol. 6, no. 3, pp. 329 - 339, 2010.
20. S. Rokicki, D. Pala, J. Paturel, and O. Sentieys, "What You Simulate Is What You Synthesize: Designing a Processor Core from C++ Specifications", *IEEE/ACM ICCAD*, 2019.
21. M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "A Free, Commercially Representative Embedded Benchmark Suite", *Int. Workshop on Workload Characterization*, 2001.
22. J. Deverge, and I. Puaut, "Safe measurement-based WCET estimation", *WCET*, 2007.