# ReMap: Reliability Management of Peak-Power-Aware Real-Time Embedded Systems through Task Replication

Amir Yeganeh-Khaksar, Mohsen Ansari, and Alireza Ejlali

**Abstract**—Increasing power densities in future technology nodes is a crucial issue in multicore platforms. As the number of cores increases in them, power budget constraints may prevent powering all cores simultaneously at full performance level. Therefore, chip manufacturers introduce a power budget constraint as Thermal Design Power (TDP) for chips. Meanwhile, multicore platforms are suitable for the implementation of fault-tolerance techniques to achieve high reliability. Task Replication is a well-known technique to tolerate transient faults. However, careless task replication may lead to significant peak power consumption. In this paper, we consider the problem of achieving a given reliability target while keeping the total power consumption under the chip TDP for a set of periodic soft real-time tasks. For this purpose, we propose a method for mapping and scheduling periodic soft real-time tasks in multicore embedded systems. The proposed method consists of three parts: (*i*) Reliability-Aware Lowest Utilization Mapping, (*ii*) Maximum-Power-Aware EDF Scheduling, and (*iii*) Reliability-and-Peak-Power-Aware Dynamic-Voltage-Frequency-Scaling. Our experiments show that our proposed method provides up to 38.4% (on average by 25%) peak power reduction compared to state-of-the-art methods.

**Index Terms**—*Reliability, Task Replication, Embedded Systems, Multicore Platforms, Thermal Design Power*

## 1 Introduction

DUE to technology scaling, the power density of multicore platforms is significantly increased [1][2][3][4]. It is an important issue because technology scaling continues to allow more transistors to be integrated onto a multicore chip while power budgets restrict the design of multicore embedded systems [1][2][3][4][5][6][30]. It is envisaged that all cores in a multicore chip cannot be simultaneously powered on at the highest performance level [1][6][32]. Due to the Thermal Design Power (TDP) constraint, system designers must decide how to use different cores in multicore platforms. According to [7], TDP is considered as "the highest sustainable power that a chip can dissipate without triggering any performance throttling mechanisms". Therefore, TDP is a power constraint that the system should meet it to operate safely without degrading the system reliability and performance [11][29][30][32]. Violating the chip TDP may automatically restart some cores or may significantly reduce their performance to prevent permanent damage [30].

Apart from the power issue, most of the embedded systems require high reliability level. It should be noted that the device-density in the chips because of the technology scaling increases the probability of fault occurrence, e.g. transient faults. Indeed, technology scaling raises the susceptibility of these systems to transient faults [8][9][10][11][24][28][32][38]. Transient faults may occur in the form of soft errors with incorrect results [8]. Since multicore systems are suitable for implementing reliability mechanisms against transient faults such as task-level redundancy [8][12][13], these mechanisms may increase peak power consumption and may cause violating the chip TDP constraint. Task replication is a quite viable option for reliability improvement in the embedded systems with multiple processing cores [8][9]. When multiple copies of the same task are executed on multiple cores, the correct execution of at least one of them is required for the system to be functional. Also, it may tolerate permanent and transient faults and creates a powerful dimension to improve the system reliability by executing multiple copies. In the following, we show how careless task replication may result in a chip TDP violation.

### 1.1 Motivational Example

For simplicity of presentation, given the homogeneous dual-core chip with 500mW of TDP that executes three tasks {$T_1$, $T_2$, $T_3$}, and their replicas. The number of tasks' replicas, their periods, and power profiles are shown in Fig. 1a. The hyperperiod of the task set is 100ms. We assume that each task has the same reliability on the different cores. It should be noted that in our ReMap scheme and in the rest of the paper, the tasks have different reliabilities even when executing on the different cores. Also, they consume different power consumption during their execution.

Two different possible schedules are shown in Fig. 1. The [8]-EM method uses task replication in such a way the reliability level of the system satisfies the reliability target and minimizes

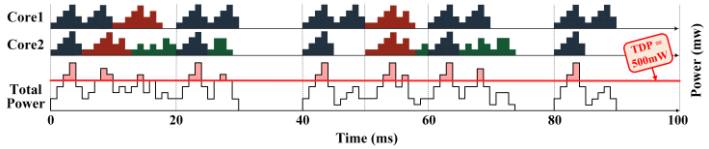
- *A. Yeganeh-Khaksar, M. Ansari, and A. Ejlali are with the Department of Computer Engineering, Sharif University of Technology, Tehran 14588, Iran (e-mail: ayeganeh@ce.sharif.edu; mansari@ce.sharif.edu; ejlali@sharif.edu).*
- *M. Ansari is also with the Karlsruhe Institute of Technology, Karlsruhe 76131, Germany (e-mail: {Mohsen.ansari@kit.edu).*

*Manuscript received 23 Dec. 2019; Revised 2 July 2020; Accepted 19 Aug. 2020; (Corresponding author: Alireza Ejlali.)*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. X.Y/X.Y*

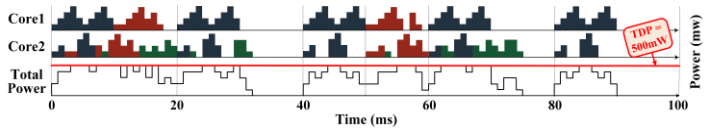
| | # Replicas | Period (ms) | Power Trace (×100mW) |
|---|---|---|---|
| $T_1$ | 2 | 20 | [1-2-3-4-2] |
| $T_2$ | 1 | 50 | [1-1-2-3-4-2-3-1] |
| $T_3$ | 0 | 50 | [1-2-1-2-1-3-2-1-3-3-1] |

| | Task1 (Red) | Task2 (Blue) | Task3 (Green) |
|---|---|---|---|
| Core1 | $T_1$ & $B_{12}$ | $B_2$ | --- |
| Core2 | $B_{11}$ | $T_2$ | $T_3$ |

(a)




(b)




(c)

Fig. 1. Peak power problem of careless task replication on a homogeneous dual-core chip, a) An example set of tasks with their mappings, b) Scheduling the tasks according to the [8]-EM, and c) Scheduling the tasks according to the ReMap scheme (Our scheme).

the energy of the system. As shown in Fig. 1, this method violates the chip-level TDP constraint in several time slots. Fig. 1c shows how ReMap schedules tasks according to the modified EDF scheduling policy to meet the TDP constraint. To achieve this, the ReMap tries to prevent overlaps of peak power of concurrently executing periodic tasks. This motivational example shows [8]-EM cannot solve the problem of meeting TDP for task replication mechanism considering a reliability target; however, the ReMap scheme improves the reliability of the tasks through task replication such that timing and peak power constraints are met. Note that in Fig. 1 $T_i$ and $B_i$ represent the main task and the replica task, respectively.

### 1.2 Our Novel Contribution

In this paper, we propose a peak-power-aware task replication mechanism (called ReMap) for a set of periodic soft real-time tasks on multicore embedded systems. The proposed method employs the task replication mechanism to satisfy the system reliability target. To satisfy a given reliability target and meet the TDP constraint, the level of replication and the voltage and frequency for each task should be determined cautiously. Indeed, the ReMap method schedules periodic soft real-time tasks on multiple cores such that satisfies timing constraints, the system reliability target, and the chip TDP constraint. For this purpose, ReMap finds the minimum level of task replication, V-F level assignment, and core allocation for each task at design time. Then, ReMap schedules tasks according to the modified EDF scheduling policy to meet the TDP constraint. At run time, ReMap detects the tasks that have executed successfully through a low-cost hardware checker and cancels the execution of their other replicas to reduce further peak power consumption and achieve more energy saving. Indeed, the ReMap method tries to prevent overlaps of peak power of concurrently executing periodic tasks such that it keeps the power consumption below the chip TDP. ReMap consists of three parts: (*i*) Reliability-Aware Lowest Utilization (RA-LU) Mapping, (*ii*) Maximum-Power-Aware Earliest-Deadline-First (MPA-EDF) Scheduling, and (*iii*) Reliability-and-Peak-Power-Aware Dynamic-Voltage-Frequency-Scaling (RPPA-DVFS) for energy management.

In order to evaluate ReMap, we ran simulations with gem5 [14] and McPAT [15] to compare ReMap and state-of-the-art methods. Our experiments show that ReMap provides up to 38.4% (on average by 25%) peak power reduction compared to state-of-the-art methods.

The rest of this paper is organized as follows. In Section 2, we review related work. We present models and assumptions in Section 3. In Section 4, ReMap is presented in detail. The experimental results are reported and discussed in Section 5. Finally, we conclude the paper in Section 6.

## 2 Related Work

The previous work related to this paper can be divided into three categories: *i*) peak power management, *ii*) average power management, and *iii*) temperature management.

### 2.1 Peak Power Management

The references [2], [4], [16], [30], and [32] can be mentioned as works that their main concern is reducing peak power consumption in compliance with timing constraints. Lee et al. [4] have presented a scheduling algorithm for real-time tasks while minimizing power consumption at the chip level. In the mentioned algorithm, no extra hardware has been used to reduce power consumption like the DVFS controller, and the algorithm only relies on scheduling tasks at the software level, and its main idea is to create limits to the parallel execution of tasks assigned to different cores. Also, it should be noted that this work does not consider fault tolerance. Munawar et al. [2] have presented a procedure that minimizes peak power by scheduling the sleep cycles of active cores in multicore systems with frame-based tasks that have a shared deadline. Lee et al. [16] have presented a scheduling algorithm for task graph models with data dependency, which prevents the violation of peak power constraint. Ansari et al. [30] have proposed a method that manages peak power overlaps between concurrently executing tasks in N-Modular Redundancy (NMR) systems while keeping the total power consumption below the chip TDP and the power consumption of each underlying core below the core TDP constraint. Recently, Ansari et al. [32] have considered a standby-sparing system where the main tasks on primary cores are scheduled by the PPA-EDF policy while the backup tasks on spare cores are scheduled by the PPA-EDL policy to meet the chip TDP constraint.

### 2.2 Average Power Management

One of the well-known techniques for reducing average power consumption is “Dynamic Voltage and Frequency Scaling (DVFS)” which works by scaling supply voltage and operating frequency [8][9][10][19][20]. The application of this technique depends on the amount of slack times on the schedule of the system. Another well-known technique is “Dynamic Power Management (DPM)” which reduces the power consumption of the whole system by power gating of inactive components [1][2].

The references [8], [17], [18], [19], and [20] have focused on reducing average power consumption by employing the mentioned techniques. Haque et al. [8] have considered the problem of achieving a given reliability target for a set of periodic real-time tasks running on a multicore system with minimum energy consumption. Their proposed method explicitly takes into account the coverage factor of the fault detection techniques and the negative impact of Dynamic Voltage Scaling (DVS) on the rate of transient faults leading to soft errors. Khavari et al. [17] have proposed a feedback-based energy management approach to estimate the execution time of real-time tasks, the relationship between past and future workloads. Ejlali et al. [18] have proposed an energy management method for frame-based tasks such that the system reliability is preserved at an acceptable level. Haque et al. [19] have presented an approach to reduce energy consumption in the form of a standby-sparing system for preemptive periodic tasks. In this work, the primary core uses DVFS and the spare core uses DPM to reduce energy consumption. Haque et al. [20] have employed the Rate Monotonic Scheduling (RMS) algorithm on a standby-sparing system for fixed priority applications, where DVFS and DPM are used on the primary core and the spare core, respectively.

### 2.3 Temperature Management

Related studies that focused on heat and temperature management can be found in [1], [21], and [22]. Pagani et al. [1] have proposed a new power budgeting concept called "Thermal Safe Power (TSP)", which provides a safe and efficient power allocation for each core based on the number of active cores in multicore systems with heterogeneous and homogeneous cores. The execution of tasks on cores while their power consumption is below TSP means that the highest chip temperature is always lower than the temperature threshold to prevent activating "Dynamic Thermal Management (DTM)". Jejurikar et al. [21] have proposed a method for allocating and scheduling tasks in hard real-time systems such that it reduces the chip temperature to a reasonable degree. Fisher et al. [22] have presented a temperature-aware scheduling mechanism that employs a method that migrates the tasks between cores and gates the power supply.

Finally, the work in high correlation to ours is [8], but the objective function of [8] is energy minimization while in this paper is peak power minimization. Indeed, [8]-EM has developed a solution for managing energy consumption without considering the chip TDP constraint. However, in this paper, we propose the MPA-EDF scheduling method that avoids overlaps of peak power of concurrent execution of periodic tasks, keeping the power consumption below the chip TDP. Therefore, the application can be executed in the system without reliability and performance degradation. Our proposed method along with the achievement of power reduction due to early completion of task and cancellation of replicas attempts to prevent overlaps of peak power of concurrently executing periodic tasks such that it always keeps the power consumption below the chip TDP constraint. The mapping mechanism of [8] is the well-known First-Fit-Decreasing (FFD) heuristic to allocate the tasks to the cores. However, in this paper, we propose the RA-LU mapping method that determines the exact number of replicas for each periodic task such that system reliability is preserved at an acceptable level. Also, RA-LU proposes the lowest utilization mapping mechanism to distribute the slack times on all cores and reduce the aging effects on the cores. In this paper, we propose the RPPA-DVFS energy management method that manages static slacks which may be generated during MPA-EDF scheduling to reduce further power and energy consumption. Since DVFS reduces the supply voltage and processing frequency, based on Eq. (2), the fault rate increases exponentially [39]. Also, increasing the execution time of tasks due to frequency reduction may result in deadline violation. Therefore, we introduced an improved DVFS technique to meet the chip TDP constraint, reliability target, and real-time constraints simultaneously in multicore embedded systems.

As discussed, the related work did not solve the problem of meeting TDP for the task replication mechanism considering a reliability target. Consequently, this paper proposes a method that exploits the task replication mechanism to improve the reliability of the tasks such that real-time and power constraints are met.

## 3 Models and Assumptions

In this section, we present our system, application, power, and fault models. We also analyze the system reliability in this section.

### 3.1 Processor Model and Workload

Our proposed system executes a set of $N$ periodic real-time tasks $T=\{\tau_1, \tau_2, \tau_3, \ldots, \tau_N\}$ with a soft deadline. It is assumed that the tasks are independent of each other and there is no data dependency between them [19]. Each task $\tau_i$ has a worst-case execution time $wc_i$ in the maximum frequency $f_{max}$. Each $\tau_i$ produces a sequence of jobs with a period $\rho_i$, and the job execution of each task must be completed before the arrival of the next job of the task. The utilization of each task $\tau_i$, as noted by $u_i$, is defined as $wc_i/\rho_i$. The $U_{total}$ is the overall utilization of the system and it is obtained from the sum of the individual tasks' utilization. The workload is executed on $M$ cores $C=\{c_1, c_2, c_3, \ldots, c_M\}$. Each core $c_j$ can execute tasks on $K$ different frequency levels from $f_{min}$ to $f_{max}$, and $F=\{f_{1=min}, f_2, \ldots, f_{K=max}\}$ is used to indicate the feasible frequency levels. In the frequency $f_k$, the core needs $wc_i/f_j$ time slots for the execution of task $\tau_i$. In this paper, $R_{target}$ is the required and desirable system's reliability for the special purpose and for its special design and usage. In order to satisfy $R_{target}$ we use task replication mechanism. In order to reach this level of reliability, the number of replicas might exceed the number of cores.

### 3.2 Power Model

Our power model is similar to [23], [25], and [27]. The power consumption of each core is made up of dynamic and static power. As it is thoroughly discussed in [26], the power consumption of a common CMOS based core at a certain time can be modeled by [33]:

$$\begin{aligned} P_{Core}(V_{dd}, f, T, t) &= P_s + P_d = P_s + \left(P_{d_{ind}} + P_{d_{dep}}\right) \\ &= V_{dd}.I_{leakage}(V_{dd}, T) + \left(P_{d_{ind}} + u_\tau(t).C_{eff}.V_{dd}^2.f\right) \end{aligned} \quad (1)$$

In Eq. (1), $u_T(t)$, $C_{eff}$, $V_{dd}$, $f$, and $I_{leakage}$ are the coefficient of core transient activity for task $\tau_i$ in the time $t$, the effective switching capacity of the core, the power supply, the core operating frequency, and the leakage current, respectively. Also, $P_{d_{ind}}$ is the frequency-independent dynamic power consumption which indicates the power consumption that the core requires for maintaining the operating mode. $P_s$ is the static power

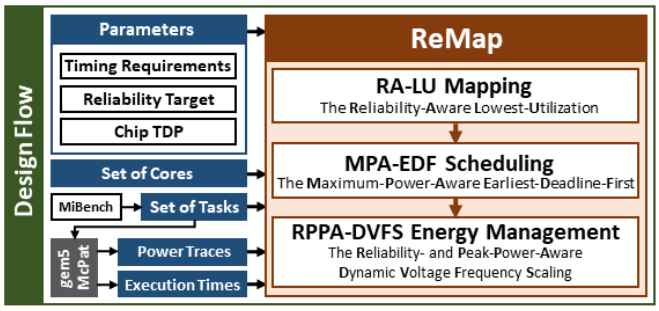



Fig. 2. The overview of the design flow of ReMap.

and it is mostly determined by the system's leakage current. It should be noted that $P_s$ dependent on the supply voltage and core temperature, *i.e.* the higher the temperature, the more the current increases. In this regard, $u_\tau(t).C_{eff}.V_{dd}^2.f$ is the dynamic power consumption which depends on the core frequency and produced due to the switching activities. $V_{dd}.I_{leakage}(V_{dd},T)$ indicates the leakage power consumption, which is mostly produced due to the leakage currents. According to the above discussion, the cores with the lower frequency and voltage levels lead to lower power consumption.

### 3.3 Reliability Model

As it was mentioned earlier, one method for increasing the system reliability is task replication. If the main tasks and their replicas are mapped on different cores of a system, the system will be able to tolerate transient to permanent faults. On the other hand, considering the occurrence of transient faults, the task replication method tries to fix the faults through time redundancy. Our fault model is similar to previous work [8]. The average failure rate of the system is dependent on the frequency of the processing core and is obtained according to Eq. (2), in which $\lambda_0$ and $d$ are the failure rate at the maximum frequency and the sensitivity to voltage changes, respectively. In this paper, we consider $d$=2 and $\lambda_0$=$10^{-7}$ faults/us. Transient faults are typically modeled as a Poisson distribution using the average failure rate of λ [8][9]. This rate significantly increases with the decrease in supply voltage for lower frequency [39]. In the situation of fixed supply voltage, the average fault rate is modeled as $\lambda(f, V) = \lambda(f) = \lambda_0 \cdot f^{b}$ [40]. It reduces linearly with the decrease of frequency (using Dynamic Frequency Scaling) due to larger safety margins in clock cycles [40]. On the other hand, in the situation of voltage scaling, the average fault rate is modeled as Eq. (2), and decreasing supply voltage increases the fault rate exponentially, which means if the frequency and supply voltage are reduced (using DVFS), the average fault rate λ increases significantly . Meanwhile, it should be noted that the DVFS technique is used in this system, and hence, Eq. (3) should be used to calculate the task's reliability with the actual execution time $t$ in different voltages and frequencies [8][9][10]:

$$\lambda(V_i) = \lambda_0 10^{\frac{V_{\max} - V_i}{d}} \tag{2}$$

$$R_i(\tau_i) = e^{-\lambda(V_i)t_i} \tag{3}$$

In Eq. 3, $\lambda(V_i)$ is given by (2). When $k$ identical copies of a task $\tau_i$ are executed on $k$ different cores, the total reliability of the task is defined as the probability of having at least one successful execution and is calculated as Eq. (4):

$$R_{total}(\tau_i) = 1 - \prod_{j=1}^{k}(1 - R_j) \tag{4}$$

Generally, the reliability of a system with $n$ tasks running by our proposed method can be calculated as:

$$R_{system} = \prod_{i=1}^{n} R_{total}(\tau_i) \tag{5}$$

It should be noted that in the paper we employ a low-cost, low power, and high accuracy hardware checker called Argus [37]. Our ReMap scheme uses Argus for fault detection on multicore embedded systems at runtime, like [33]. Runtime execution increases on average by 3.9% using Argus [37], and we consider this overhead within the worst-case execution time of each task. Indeed, ReMap detects the tasks that have executed successfully using Argus and cancels the execution of their other replicas to reduce further peak power consumption and achieve more energy saving.

## 4 Our Proposed Method

### 4.1 Problem Definition and System Overview

In this paper, one of the constraints is meeting the reliability target. Since Eq. 5 is an exponential equation, the mentioned problem is convex [8][33][36]. The convex formulated problem can be solved by the available convex (CVX) solvers, and it is categorized as an NP-Complete problem [1][2][4]. On the other hand, the complexity of such problems may increase exponentially with the increase of problem size, e.g., with the number of tasks, cores, and frequency levels. Therefore, we have proposed a heuristic-based method to provide an effective solution for the presented problem.

An overview of design flow of ReMap is shown in Fig. 2. In our framework, the features of the ARM Cortex-A processors in gem5 and McPAT tools are used to generate power traces. To achieve the purpose of this study, we provide a system-level peak power management method. This method includes three phases: mapping, scheduling, and energy management. The Reliability-Aware Lowest-Utilization (RA-LU) task mapping mechanism, in which mapping operations to the cores are done with respect to their utilization and with reliability awareness, is discussed in Subsection 4.2. In Subsection B.2, the Maximum-Power-Aware Earliest-Deadline-First (MPA-EDF) scheduling mechanism that schedules the tasks according to their deadline and peak power consumption is explained. The main purpose of MPA-EDF is offline task scheduling in a way that does not violate the TDP. The Reliability-and-Peak-Power-Aware Dynamic-Voltage-Frequency-Scaling (RPPA-DVFS) energy management mechanism will also be introduced in the Subsection B.3), which describes the reduction of average power consumption by DVFS technique and with the awareness of peak power consumption and acceptable system reliability.

### 4.2 The Proposed Method Discussion

In this section, we express our method's algorithms. The notation of the parameters used in our algorithms is described in Table 1.

#### I) RA-LU Mapping

Table 1. The Notation of the Parameters

| Notation | Description |
|---|---|
| T | A set of tasks |
| C | A set of cores |
| $R_{\text{target}}$ | The reliability target of the system |
| $R_{\text{total}}$ | The reliability of the system |
| $\tau_{k,l}$ | The $l^{\text{th}}$ replica of a task $\tau_k$ |
| $j^i_{\tau_{k,l}}$ | The $i^{\text{th}}$ job of the task $\tau_{k,l}$ |
| $j^i_{\tau_{k,l}}.f$ | The frequency of the job $j^i_{\tau_{k,l}}$ |
| $j^i_{\tau_{k,l}}.rt$ | The release time of the job $j^i_{\tau_{k,l}}$ |
| $j^i_{\tau_{k,l}}.ex$ | The execution time of the job $j^i_{\tau_{k,l}}$ |
| $j^i_{\tau_{k,l}}.dl$ | The deadline of the job $j^i_{\tau_{k,l}}$ |
| $j^i_{\tau_{k,l}}.li$ | The last scheduled time of the job $j^i_{\tau_{k,l}}$ |
| $RQ$ | A ready queue of task instances (jobs) |
| $ST$ | A set of slack times |

**Algorithm 1. The task mapping mechanism (RA-LU) of ReMap**

**Input:** set of tasks (T), set of cores (C), $R_{\text{target}}$
**Output:** The tasks mapping on each core

```
start RA-LU
1: R_total ← 1;  T̂ ← T; Construct the TCFR table;
2: errorFlag ← false;  assignFlag ← false;
3: while (∃ τ ∈ T̂) and !errorFlag
4:    c ← findMinU(C);
5:    unmark all tasks in T̂;  assignFlag ← false;
6:    while (∃ τ ∈ (T̂ − {marked tasks}))
7:       τ ← findMaxR(T̂ − {marked tasks}, {c}, f_max);
8:       if  !assignTask(τ, c) then
9:          omit task from T̂;  assignFlag ← true;  break;
10:      else  mark task τ;
11:   if  !assignFlag then  errorFlag ← true;
12: update R_total;  update reliability of all tasks;  T̂ ← T;
13: while (R_total < R_target and !errorFlag)
14:   if (∃ τ ∈ T̂) then
15:      τ ← findMinR(T̂, C, f_max);
16:      mark cores with task τ in C;  assignFlag ← false;
17:      while (∃ c ∈ (C − {marked cores}))
18:         c ← findMinU(C − {marked cores});
19:         if  !assignTask(τ, c) then assignFlag ← true; break;
20:         else  mark core c in C;
21:      if !assignFlag then
22:         c ← findMinU(C);
23:         if  !assignTask(τ, c) then
24:            omit task from T̂;
25:            update R_total; update reliability of all tasks;
26:         else
27:            errorFlag ← true;
28:   else
29:      errorFlag ← true;
30: if !errorFlag then
31:   while (∃ τ ∈ T)
32:      while (∃ j ∈ τ)
33:         j^i_{τ_{k,l}}.li ← j^i_{τ_{k,l}}.rt; j^i_{τ_{k,l}}.f ← f_max;
34:   return true;
35: else return false;
end RA-LU
```

Algorithm 1 shows the pseudo-code of the task mapping mechanism of our ReMap method that receives reliability target, sets of tasks and cores to create tasks' maps for the scheduling part. At first, Algorithm 1 initializes the reliability of the system to 1 and makes a copy of the set of tasks, and also constructs the TCFR table in line 1. In the TCFR table, the reliability values for all tasks are given for different cores, and voltages/frequencies. In TCFR, $R_{ijk}$ is the reliability value of the task $\tau_i$ on the core $c_j$ with the frequency $f_k$ obtained using Eq. (3). In line 2, two flags, *assignFlag* and *errorFlag*, are defined and initialized to *false* for the algorithm. Next, the algorithm iterates as long as there is a task in the set $\hat{T}$ (lines 3-11). In line 4, the function *findMinU*(C) returns the core with the lowest utilization from the set of core C. In the next step, when a task is mapped to the selected core, the algorithm marks the task and changes the *assignFlag* to *true*. This is because, before everything, the algorithm should unmark all tasks in $\hat{T}$ and change *assignFlag* to *false*. Now, the algorithm iterates until there is a task in the set $\hat{T}$ − {marked tasks} (lines 6-10),. Indeed, the algorithm investigates all the unmarked tasks to map a core with the lowest utilization. The function *findMaxR*(T,C,*f*) returns the task with the highest reliability from the task set T for the frequency *f* and the set of core C (according to the TCFR table). In line 7, considering the highest frequency, the algorithm maps a task in the set $\hat{T}$ − {marked tasks} with the highest reliability to the core with the lowest utilization. In line 8, the function *assignTask*($\tau$, *c*) is used to map the task $\tau$ to the core *c*. If the mapping operation is successful, it returns a zero value. Otherwise (due to the high utilization), a non-zero value is returned. If the mapping operation is successful (line 9), the algorithm omits the selected task from set $\hat{T}$ and changes *assignFlag* to *true*, and then the algorithm goes to line 11. Otherwise (line 10) the selected task is marked and the **While** loop is repeated. After the **While** loop, if *assignFlag* remains *false*, which means none of the initial tasks could be mapped to the core with the lowest utilization, the algorithm changes *errorFlag* to *true* and returns *false* (line 35). In line 12, the reliability of all tasks and the reliability of the system are updated, and also another copy of the set of tasks is considered. Up to this line, the algorithm attempts to do the initial task mapping. In the following sequence (lines 13-29), the algorithm iterates and attempts to replicate tasks to satisfy the reliability target. Similar to the *findMaxR*(T,C,*f*) function, the *findMinR*(T,C,*f*) function returns the task with the lowest reliability from the task set T for the frequency *f* and the set of core C. In line 15, considering the highest frequency and the set of core C, the algorithm selects a task $\tau$ in the set of tasks $\hat{T}$ with the lowest reliability.

Since the algorithm attempts to distribute replicas between different cores to tolerate both transient and permanent faults, all the cores with at least one replica of the task $\tau$ are marked in line 16. In the inner **While** loop (lines 17-20), the algorithm iterates for each core in the set C − {marked cores}, and then a core with the lowest utilization is selected from C − {marked cores} in line 18. If mapping operation is successful (line 19), the algorithm changes *assignFlag* to *true* and goes to line 21. Otherwise the selected core is marked in line 20 and the **While** loop is repeated. After doing inner **While** loop (lines 17-20), if *assignFlag* remains *false*, the core with the lowest utilization is selected instantly for mapping in line 22. If mapping operation to the core with the lowest utilization is successful (line 24-25), the algorithm updates the reliability of all tasks and the reliability of the system and goes to line 13. Otherwise, replicas may increase so much due to the low-reliability level of each task or the high level of the desired system reliability, and they cannot be mapped as the result of timing and utilization constraints. In

**Algorithm 2. The task scheduling mechanism (MPA-EDF) of our ReMap method**

**Input:** set of tasks (T), set of cores (C), TDP
**Output:** The tasks scheduling on each core

**start MPA-EDF**
1: $RQ \leftarrow$ array of all jobs of all tasks;
2: *errorFlag* ← false; *assignFlag* ← false;
3: **while** (∃ $j \in RQ$) and !*errorFlag*
4: $j^i_{\tau_{k,l}} \leftarrow getFirstJob(RQ)$; *assignFlag* ← false;
5: **while** ($j^i_{\tau_{k,l}}.li \leq j^i_{\tau_{k,l}}.dl$)
6: **if** !$isEmpty(c_{\tau_{k,l}}.time(j^i_{\tau_{k,l}}.li))$ **then**
7: **if** $j^i_{\tau_{k,l}}.time(0).pwr + allPower(j^i_{\tau_{k,l}}.li) \leq$ TDP **then**
8: $j^i_{\tau_{k,l}}.ex \leftarrow j^i_{\tau_{k,l}}.ex - 1$; schedule first time slot of $j^i_{\tau_{k,l}}$;
9: **if** ! $j^i_{\tau_{k,l}}.ex$ **then** omit $j^i_{\tau_{k,l}}$ from RQ;
10: *assignFlag* ← true; break;
11: $j^i_{\tau_{k,l}}.li \leftarrow j^i_{\tau_{k,l}}.li + 1$;
12: **if** !*assignFlag* **then** *errorFlag* ← true; break;
13: **if** !*errorFlag* **then** return true;
14: **else** return false;
**end MPA-EDF**

this situation, the algorithm changes *errorFlag* to *true* and goes to line 35, and then encounters an error. Now, if *errorFlag* remains *false*, $j^i_{\tau_{k,l}}.li$ and $j^i_{\tau_{k,l}}.f$ (See Table 1) for each instance (job) of each task are initialized. Finally, the algorithm returns *true* which means the mapping of the tasks on the cores is ready for scheduling part of our ReMap method.

## II) MPA-EDF Scheduling

Algorithm 2 shows the pseudo-code of the task scheduling mechanism of our ReMap method that receives TDP, task-to-core-assignment, sets of tasks and cores to create tasks' scheduling. In line 1, the algorithm initializes a queue *RQ* (See Table 1) with all instances of all tasks. In line 2, the flags *assignFlag* and *errorFlag* are initialized to *false*. The algorithm iterates until there is an instance in the *RQ* (lines 3-14). The *getFirstJob*(*RQ*) function returns the job with the highest execution priority (which means the earliest deadline and the highest peak power consumption) from the *RQ*. In line 4, the algorithm selects the first instance by the *getFirstJob*(*RQ*) function for execution and changes *assignFlag* to *false*. The inner **While** loop (lines 5-11) iterates as long as the last scheduled time of the selected job is smaller than its deadline. The function *isEmpty*(*c.time*(*t*)) is used to check whether the time slot *t* of core *c* is empty or not. If the mentioned time slot is empty, it returns a zero value; otherwise, a non-zero value is returned. The *allPower*(*index*) function returns the total power consumption of all cores at the time slot *index*. In line 7, the algorithm adds the power consumption of the first time slot ($j^i_{\tau_{k,l}}.time(0).pwr$) to the total power consumption in the last scheduled time ($allPower(j^i_{\tau_{k,l}}.li)$) of selected instance, and then compares it with TDP constraint. If the total power consumption meets the TDP constraint, lines 8 to 11 are executed, otherwise, only line 11 is executed. In line 8, the algorithm decreases the execution time and schedules the first time slot of the selected instance. When each task instance is successfully finished during the algorithm (line 9), it is omitted from *RQ*. In line 10, the algorithm changes *assignFlag* to *true* and goes to line 12. At every iteration, the last scheduled time of the instance increases in line 11. In line 12, if *assignFlag* remains *false* due to the TDP and timing constraints, the algorithm changes *errorFlag* to *true*, and then goes to line 14 and encounters an *error*. Otherwise, the algorithm returns *true* which means the tasks scheduling on each core is ready.

**Algorithm 3. The task energy management mechanism (RPPA-DVFS) of our ReMap method**

**Input:** set of tasks (T), set of cores (C), TDP, available frequencies (F), $R_{target}$

**start RPPA-DVFS**
1: $ST \leftarrow$ array of all slack times of all cores;
2: *assignFreqFlag* ← false;
3: **while** (∃ *st* ∈ *ST*)
4: $st \leftarrow getFirstSlack(ST)$;
5: $c.st \leftarrow$ core of *st*;
6: $J \leftarrow$ array of all the jobs of *c.st*;
7: **while** (∃ *j* ∈ *J*)
8: **if** (*j.rt* ≥ *st.end*) or (*j.dl* ≤ *st.start*) **then** omit *j* from *J*;
9: **while** (∃ *j* ∈ *J*)
10: $j^i_{\tau_{k,l}} \leftarrow getFirstJob(J)$;
11: $f_x \leftarrow f_{min}$; *assignFreqFlag* ← false;
12: **while** $f_x \leq f_{max}$
13: **if** $replica(j^i_{\tau_{k,l}}, f_x, R_{total}, R_{target})$ **then**
14: $j^{i_r}_{\tau_{k,l}} \leftarrow$ the replication of $j^i_{\tau_{k,l}}$;
15: add $j^{i_r}_{\tau_{k,l}}$ to *J*;
16: $j^{i_r}_{\tau_{k,l}}.f \leftarrow f_{max}$;
17: $j^{i_r}_{\tau_{k,l}}.rt \leftarrow j^i_{\tau_{k,l}}.li + 1$;
18: $j^{i_r}_{\tau_{k,l}}.li \leftarrow j^{i_r}_{\tau_{k,l}}.rt$;
19: $j^{i_r}_{\tau_{k,l}}.ex \leftarrow j^i_{\tau_{k,l}}.ex$;
20: **if** MPA_EDF(J, C, TDP) **then**
21: *assignFreqFlag* ← true; break;
22: **else** $j^i_{\tau_{k,l}}.f \leftarrow f_{max}$; omit $j^{i_r}_{\tau_{k,l}}$ from *J*; $f_x \leftarrow f_{x+1}$;
23: **if** !*assignFreqFlag* **then** omit $j^i_{\tau_{k,l}}$ from *J*;
24: **else** break;
25: **if** (∄ *j* ∈ *J*) and !*assignFreqFlag* **then** omit *st* from *ST*;
26: update *ST*;
**end RPPA-DVFS**

## III) RPPA-DVFS Energy Management

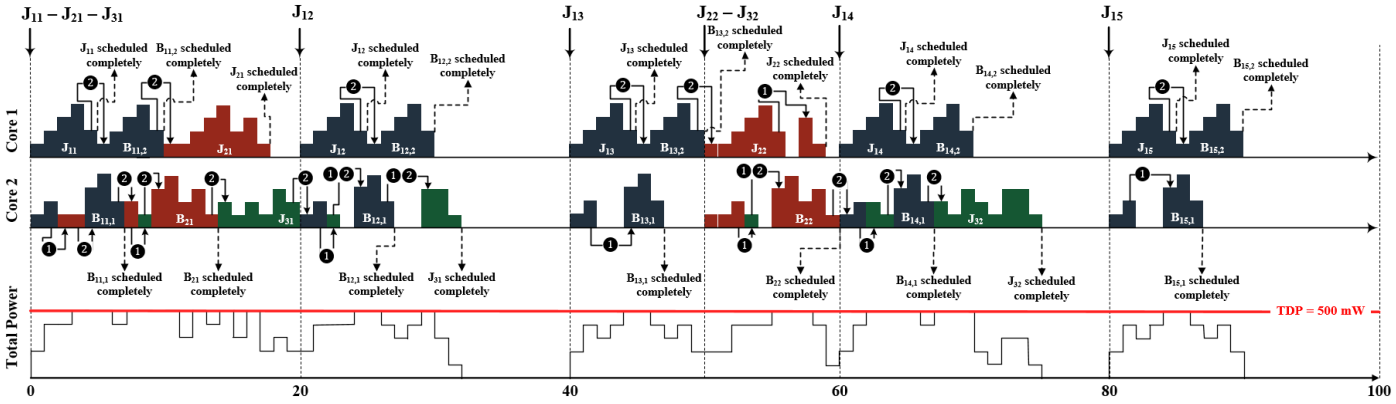
Algorithm 3 shows the pseudo-code of the energy management mechanism of our ReMap method that receives TDP, the set of available frequencies, tasks, and cores and the system reliability target. In line 1, the algorithm initializes *ST* (See Table 1) with all slack times of all cores. The flag *assignFreqFlag* is initialized to *false*. Next, the algorithm iterates until there is a slack time in the *ST* (lines 3-26). The *getFirstSlack*(*ST*) function returns the slack time with the highest priority (which means the longest length) called *st* from the *ST*. In line 4 and 5, the core corresponding to the selected slack time, called *c.st*, and an array of all task instances corresponding to the selected core, called *J*, are selected. In the first nested **While** loop (lines 7-8), all the instances with the release time *j.rt* greater than the end of slack time *st.end* or the deadline *j.dl* smaller than the start of slack time *st.start* are omitted from *J*. In the second nested **While** loop (lines 9-26), the algorithm iterates as long as there is an instance in the *J*. In line 10, the *getFirstJob*(*J*) function returns an instance with the highest execution priority (which means the earliest deadline and

| # | Reliability | #Replicas before *RA-LU* | #Replicas after *RA-LU* | Period | Power Trace (×100mW) |
|---|---|---|---|---|---|
| $T_1$ | 0.999999721 | 0 | 2 | 20 | [1-2-3-4-2] |
| $T_2$ | 0.999999865 | 0 | 1 | 50 | [1-1-2-3-4-2-3-1] |
| $T_3$ | 0.999999917 | 0 | 0 | 50 | [1-2-1-2-1-3-2-1-3-3-1] |
| **System Reliability** | | 0.999999590 | 0.999999911 | --- | |

(a)

| Core | Tasks and their replicas mapping after *RA-LU* | | |
|---|---|---|---|
| | Task1 | Task2 | Task3 |
| Core1 | $T_1$ & $B_{12}$ | $B_2$ | --- |
| Core2 | $B_{11}$ | $T_2$ | $T_3$ |

(b)




(c)

Fig. 3. An example of how our proposed method works on a homogeneous dual-core system with TDP=500mW and $R_{target}$=1-10$^{-7}$. a-b) An example task set before and after the implementation of the RA-LU with their number of replicas, power traces, and core mapping, c) The MPA-EDF scheduling.

the highest peak power) from *J*. Before any frequency reduction, the algorithm considers the lowest frequency $f_{min}$ for applying to the selected instance and also changes *assignFreqFlag* to *false* in line 11. It should be noted that $f_x$ is defined as the lower frequency applied to the task instance. The algorithm attempts (lines 12-19) to apply the lowest possible frequency in a way that does not violate TDP in the next slots. The **While** loop (lines 12-22) iterates until $f_x$ reaches $f_{max}$. The function *replica*() specifies whether by the decrease of the instance's operating frequency a replica is needed or not (because it is possible the system reliability does not satisfy the reliability target). If a replica is required, an additional replica is added (line 14-16) and scheduled (lines 16-19). This is because if the main job is executed successfully, the execution of its replicas can be canceled. The function *MPA-EDF*(J, C, TDP) (based on Algorithm 2) checks whether scheduling operations with lower frequency can be performed. If this function returns *true*, in line 21, the algorithm changes *assignFreqFlag* to *true* and goes to line 25. Otherwise, in line 22, the algorithm increases the frequency of the task instance to $f_{max}$ and omits the additional replica from *J*, and then increases $f_x$ to the next higher level. In line 23, if *assignFreqFlag* remains *false*, task instance is omitted from *J*, which means the algorithm cannot apply lower frequency to the selected task instance. Similarly in line 25, if *assignFreqFlag* remains *false* and there is no task instance in *J*, the selected slack time is omitted from *ST*, which means the algorithm cannot apply lower frequency to any task instances for the selected slack time. Finally, the algorithm updates *ST* (line 26) and repeats the above mentioned process until there is slack time in *ST*.

### 4.3 Analysis of Time Complexity

In the proposed algorithms, suppose that *N* is the number of all tasks, *M* is the number of cores, *h* is the total time slots, and *l* is the number of voltage-frequency levels. The main computation of algorithms is performed to map and schedule all tasks and then putting them into a max-heap. Therefore, for *N* tasks, *M* cores, and *h* time slots, building the max-heap is performed in *O(M×N)*. The first algorithm (*RA-LU Mapping*) iterates for *O(M×N)* times. The second algorithm (*MPA-EDF Scheduling*) iterates for *O(M×N×h)* times. The third algorithm (*RPPA-DVFS Energy Management*) iterates for *O(M×N×l)* times. Therefore, the order of the algorithm is *max{O(M×N), O(M×N×l), O(M×N×l)}*.

### 4.4 An Example of Our Proposed Method

Let's consider the proposed approach by presenting a more detailed example. In this example, three periodic tasks of $T_1$, $T_2$, and $T_3$ are considered to be scheduled on a dual-core system with TDP=500mW. The number of replicas, the worst-case execution time, the period, and the power trace of the three mentioned tasks are shown in Fig. 3a. In order to determine the number of replicas and map the main and replica tasks, we ran Algorithm 1. Therefore, $T_1$, $T_2$, and $T_3$ have 2, 1 and zero replicas, respectively, according to Fig. 3b. Then, Algorithm 1 maps $T_1$ and the first replica of $T_1$ to Core1 and Core2, respectively. Also, the second replica of $T_1$ is mapped to Core1. In order to balance the utilization of cores, Algorithm 1 maps $T_2$ and its replica to Core2 and Core1, respectively. Meanwhile, $T_3$ is mapped to Core2. In the following example, we have considered $B_{ij}$ as a replica of $J_{ij}$. At *t*=0ms, the first jobs of all the tasks are considered to be scheduled on the cores. $J_{11}$, $J_{21}$, and $B_{11,2}$ (the second replica of $J_{11}$) are ready to be scheduled on Core1. Also, $J_{31}$, $B_{21}$ (the replica of $J_{21}$), and $B_{11,1}$ (the first replica of $J_{11}$) are ready to be scheduled on Core2. At first, $J_{11}$ is selected to be scheduled because its deadline is closer than the deadlines of other ready jobs. The first time slot of $J_{11}$ is scheduled on Core1 in the time slot [0ms, 1ms]. Out of three jobs $B_{11,1}$, $B_{21}$, and $J_{31}$ mapped to Core2, $B_{11,1}$ is selected to be scheduled. These decisions are also established in the time slot [1ms, 2ms]. In the third time

slot [2ms, 3ms], on the Core1, task $J_{11}$ is considered to be scheduled, while on Core2, $B_{11,2}$ cannot be scheduled because the peak power consumption exceeds TDP. At $t$=20ms, when $J_{12}$ is released since $B_{12,1}$ (first replica of $J_{12}$) has a closer deadline relative to $J_{31}$, it is decided to be scheduled earlier than $J_{31}$. On Core1, $J_{12}$ is also selected to be scheduled because all the jobs released at $t$=0ms are scheduled before $t$=20ms, and Core1 is in the idle state.

In Fig. 3c, the arrows marked with the number ❶ represent the times switched between tasks due to TDP violation. For example, at $t$=8ms, if $B_{21}$ is scheduled on Core2, the TDP constraint is violated, Therefore $J_{31}$ is scheduled at this time. As another example at $t$=42ms, since the execution of $B_{13,1}$ violates TDP and there is no other job in the queue to be scheduled on Core2, $B_{13,1}$ is shifted until its execution does not violate TDP. The other arrows marked with the number ❷ represent the times when a task with a closer deadline should be scheduled for execution. For example, at $t$=64ms, $B_{14,2}$ should be scheduled due to the closest deadline relative to $J_{32}$. At $t$=62ms, since the scheduling of $B_{14,2}$ would have violated TDP, $J_{32}$ is considered to be scheduled instead of $B_{14,1}$. The arrows marked with numbers ❶❷ indicate that a job is selected to be scheduled due to the closest deadline and also is selected to be shifted due to TDP violation. For example, at $t$=54ms, since $B_{22}$ has a closer deadline compared to $J_{32}$, it is considered to be scheduled but because of TDP violation, it must be shifted to the next time slot. At $t$=56ms, between $J_{22}$ and $B_{22}$ which have the highest priority on their corresponding cores, $B_{22}$ is selected to be scheduled on Core2 because of higher priority than $J_{22}$. Then, $J_{22}$ on the Core1 is shifted to $t$=57ms because of TDP violation.

As previously stated in Section IV and Subsection B.2, Jobs with closer deadlines and jobs with higher peak power consumption compared to other jobs with the same deadline have the highest priority for scheduling. For example, at $t$=20ms, when $B_{12,1}$ is released, it is immediately selected to be scheduled because it has a deadline ahead of $J_{31}$. Also, at $t$=14ms, when $B_{21}$ is fully scheduled, it is removed from the queue, and $J_{31}$ is considered to be scheduled. It should be noted that at this time, between $J_{21}$ and $J_{31}$, since $J_{21}$ has a higher peak power, at first $J_{21}$ and then $J_{31}$ are considered to be scheduled.

# 5 Results and Discussion

## 5.1 Experimental Setup

In this section, we investigated the impact of our proposed method by simulating various tasks based on the MiBench benchmark suite [31] and on the 4-core, 8-core, and 16-core systems. Firstly, we clarify how we have produced our tasks' sets and their power traces. Then, the comparison between the proposed method and state-of-the-art methods are discussed. We exploit ARM processors in our evaluations because this kind of processor is widely used in embedded systems [32]. It is also assumed that the system supports core-level DVFS, and there are 6 different frequency/voltage levels from [0.85Volt, 1GHz] to [1.1Volt, 2GHz]. Various applications from MiBench benchmark suite for different inputs are generated in gem5 and McPAT integrated simulator. We have generated more than 1000 random inputs to achieve power trace, minimum and maximum power consumption, suitable voltage/frequency levels, execution time, and energy for each application.

Some studies focused on cases of reliability and energy, but they did not pay attention to peak power consumption. Some other studies have also managed reliability in the form of hardware redundancy with respect to peak power consumption. One of the research studies that has been done to increase and maintain reliability and with the goal of minimizing energy is [8]-EM. In the [8]-EM, as discussed earlier in Section II, tasks are replicated in such a way the reliability level of the system reaches the acceptable reliability target and minimizes the energy of the system. Both of the proposed method and method [8]-EM use task replication to satisfy the reliability target of the system, therefore, the peak power of these two methods are compared together.

## 5.2 Result Discussion

We evaluate our proposed method in the realistic scenario in which the fault rate is based on Eq. 2. The peak power consumption analysis provides the best conditions for comparing our proposed method and [8]-EM. Fig. 4 shows the comparison of peak power consumption in [8]-EM and our proposed method, ReMap. It shows that the peak power of ReMap is always less than that of the method [8]-EM. In Fig. 4, the dashed line expresses the value of TDP. As can be seen, the method [8]-EM violates TDP constraint in all conditions. In this experiment (Fig. 4a to Fig. 4c), for the per-core utilization of 0.5, an identical random task set with the same inputs is given to execute on the 16-core system, and the reliability targets are $1\text{-}10^{-9}$, $1\text{-}10^{-7}$, and $1\text{-}10^{-}$

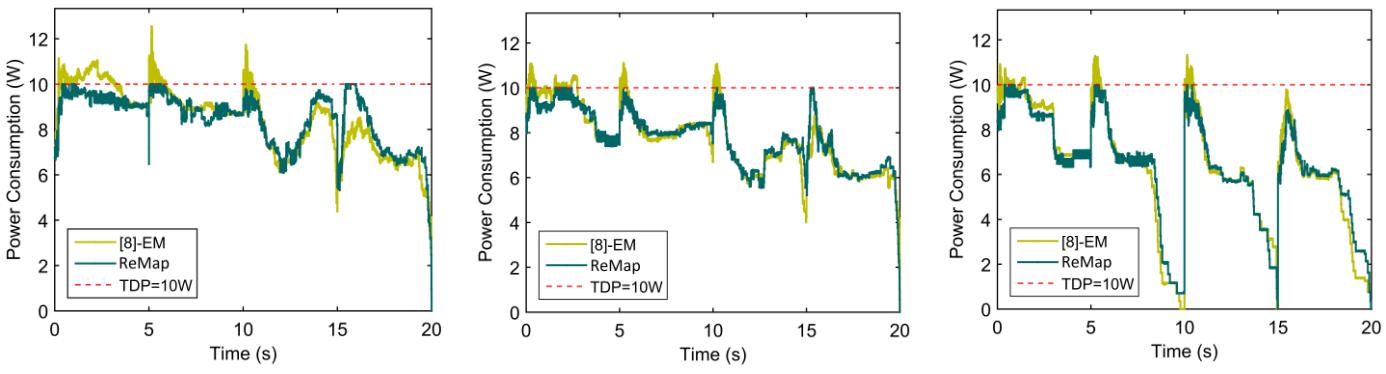



Fig. 4. Power consumption profile in the worst-case scenario on a 16-core system, a) $R_{target}=1\text{-}10^{-9}$, b) $R_{target}=1\text{-}10^{-7}$, c) $R_{target}=1\text{-}10^{-5}$.

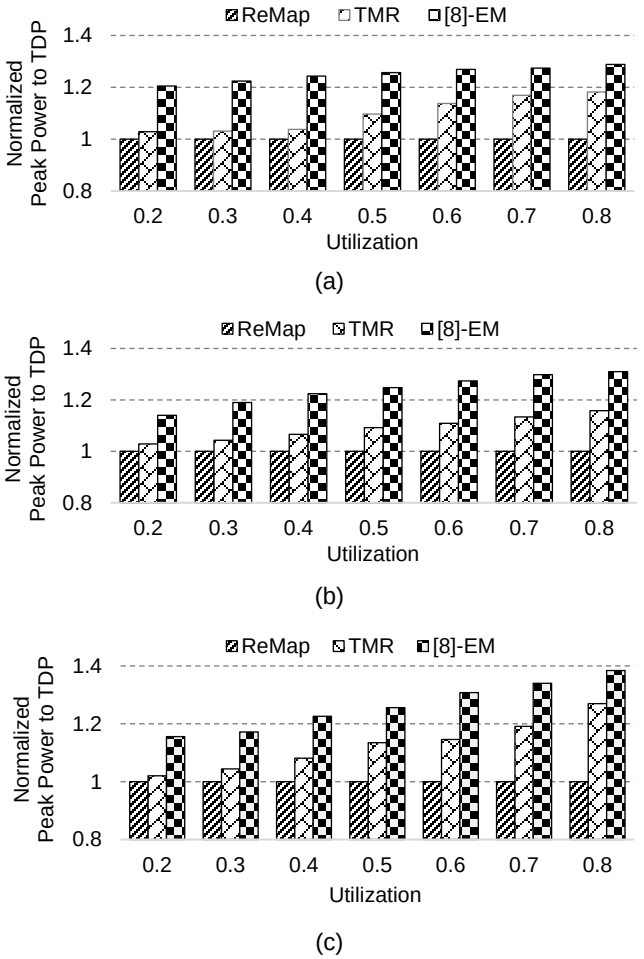



Fig. 5. Normalized peak power consumption to TDP for the realistic execution on homogeneous multicore systems with $R_{target}$=1-10$^{-7}$ and core utilization of 0.5, a) #cores=16, b) #cores=8, and c) #cores=4.

$^{5}$, respectively. As with the above experiment, 1000 other task sets were performed for each utilization between 0.2 to 0.8 and the number of cores was 4, 8, and 16. Then, average results are presented in Fig. 5. These simulations' results indicate that the proposed method provides up to 38.4% and on average by 25% peak power reduction compared to conventional triple modular redundancy (TMR) [30] and [8]-EM. It should be noted that the objective function of [8] is energy minimization while in this paper is peak power minimization. Indeed, [8]-EM has developed a solution for managing energy consumption without considering the chip TDP constraint. However, in this paper, we propose the peak-power-aware mapping and scheduling method that avoids overlaps of peak power of concurrent execution of periodic tasks, keeping the power consumption below the chip TDP. Our proposed method along with the achievement of power reduction due to early completion of task and cancellation of replicas attempts to prevent overlaps of peak power of concurrently executing periodic tasks such that it always keeps the power consumption below the chip TDP constraint.

We have also compared our proposed method with the following real-time scheduling algorithms for schedulability analysis:

- LST: It is an optimal algorithm with the dynamic priority assignment that schedules tasks based on laxity. In this algorithm, the task with the shortest laxity gets the highest priority [34].
- RM: This algorithm is used for scheduling independent real-time tasks. It schedules tasks based on their periods with static priority assignment. In this algorithm, the task with the shortest period gets the highest priority [35].

Similar to our MPA-EDF policy, MPA-LST and MPA-RM are improved to meet the power constraints for the models of above scheduling algorithms in comparison with the proposed method. Due to the static priority assignment, the implementation of MPA-RM is much simpler than MPA-EDF and MPA-LST. As shown in Fig. 6a, in terms of utilization, static scheduling algorithm, MPA-RM, performed worse than dynamic priority scheduling algorithms, MPA-EDF and MPA-LST. Whereas MPA-EDF and MPA-LST are the same or very similar in terms of schedulability. Many context switches happen in the MPA-LST due to its scheduling criteria.

As the final discussion, we discuss the schedulability of the proposed method and [8]-EM in the worst-case scenario on a 16-core system for different reliability targets. In this scenario, all tasks are executed. To demonstrate this, we generated 1000 task sets and repeated the simulations for several utilizations. Fig. 6b shows the schedulability for our proposed method and [8]-EM. The results show that our method meets all constraints simultaneously on average by 43.02% while the [8]-EM meets the reliability and real-time constraints on average by 37.33%, whereas it cannot satisfy TDP constraint. Therefore, ReMap is

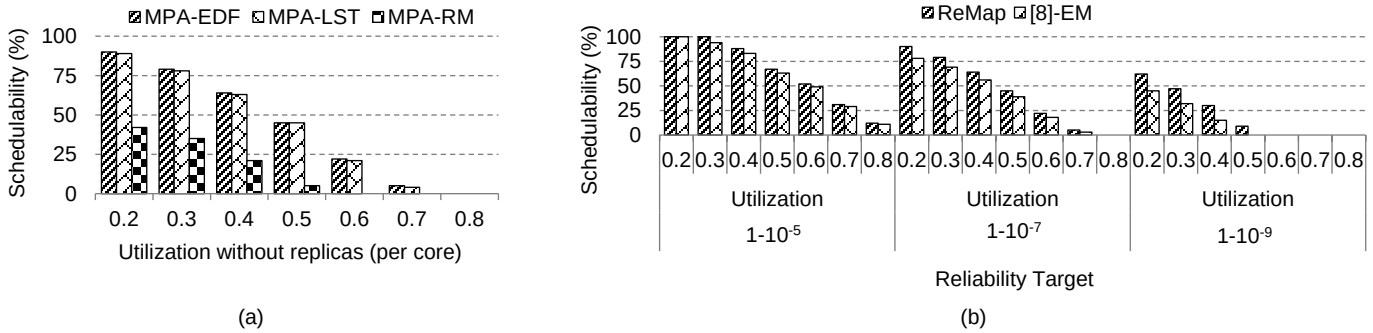



Fig. 6. a) Schedulability comparison of MPA-EDF and the improved LST and RM real-time scheduling algorithms, b) Schedulability in the worst-case scenario on a 16-core system for different reliability targets.

more efficient than [8]-EM for meeting the timing and TDP constraints simultaneously.

## 6 Conclusion and Future Work

In this paper, we have considered two main objectives in designing real-time multicore embedded systems such as reliability and peak power consumption. To achieve these two objectives, a method is proposed for creating replicas of periodic real-time tasks. Our ReMap method consists of three phases: RA-LU Mapping, MPA-EDF Scheduling, and RPPA-DVFS Energy Management, which are executed sequentially and in the event of successful execution of the previous phase. Due to the fact that they rarely occur in normal mode, and tasks are usually completed sooner than their worst-case execution time, successful execution of tasks has taken place earlier and the execution of replicas is canceled. We compared the proposed method with state-of-the-art methods and the results indicate that the peak power is reduced by 38.4% at best and 25% on average.

Unlike periodic tasks, aperiodic tasks are event-driven and have irregular arrival times. Due to the nature of aperiodic tasks, a method with a low order of complexity should be exploited at runtime to guarantee that all the constraints are met. We will explore the aperiodic task model in our future work.

## References


[1] S. Pagani, H. Khdr, J. Chen, M. Shafique, M. Li, and J. Henkel, "Thermal safe power (tsp): Efficient power budgeting for heterogeneous manycore systems in dark silicon," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 147–162, Jan 2017.

[2] W. Munawar, H. Khdr, S. Pagani, M. Shafique, J. Chen, and J. Henkel, "Peak power management for scheduling real-time tasks on heterogeneous many-core systems," in *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 200–209, Dec 2014.

[3] S. Pagani, H. Khdr, W. Munawar, J. Chen, M. Shafique, M. Li, and J. Henkel, "Tsp: Thermal safe power-efficient power budgeting for many-core systems in dark silicon," in *2014 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 1–10, Oct 2014.

[4] J. Lee, B. Yun, and K. G. Shin, "Reducing peak power consumption in multicore systems without violating real-time constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 1024–1033, April 2014.

[5] S. Pagani, A. Pathania, M. Shafique, J. Chen, and J. Henkel, "Energy efficiency for clustered heterogeneous multicores," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1315–1330, May 2017.

[6] M. Ansari, M. Salehi, S. Safari, A. Ejlali, and M. Shafique, "Peak-Power-Aware Primary-Backup Technique for Efficient Fault-Tolerance in Multicore Embedded Systems," *IEEE Access*, vol. 8, pp. 142843-142857, 2020.

[7] Intel Corporation, "Dual-core intel xeon processor 5100 series datasheet, revision 003," August 2007.

[8] M. A. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 813–825, March 2017.

[9] M. Salehi, A. Ejlali, and B. M. Al-Hashimi, "Two-phase low energy n-modular redundancy for hard real-time multicore systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1497–1510, May 2016.

[10] F. R. Poursafaei S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Offline replication and online energy management for hard real-time multicore systems," in *2015 CSI Symposium on RealTime and Embedded Systems and Technologies (RTEST)*, pp. 1–7, Oct 2015.

[11] M. Ansari, M. Pasandideh, J. Saber-Latibari, and A. Ejlali, "Meeting Thermal Safe Power in Fault-Tolerant Heterogeneous Embedded Systems," *IEEE Embedded Systems Letters*, vol. 12, no. 1, pp. 29-32, 2020.

[12] D. K. Pradhan, Ed., *Fault-tolerant Computer System Design*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[13] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[14] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[15] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 469–480, Dec 2009.

[16] B. Lee, J. Kim, Y. Jeung, and J. Chong, "Peak power reduction methodology for multicore systems," in *2010 International SoC Design Conference*, pp. 233–235, Nov 2010.

[17] M. K. Tavana, M. Salehi, and A. Ejlali, "Feedback-based energy management in a standby-sparing scheme for hard real-time systems," in *2011 IEEE 32nd Real-Time Systems Symposium*, pp. 349–356, Nov 2011.

[18] A. Ejlali, B. M. Al-Hashimi, and P. Eles, "A standby-sparing technique with low energy-overhead for faulttolerant hard real-time systems," in *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '09. New York, NY, USA: ACM, pp. 193–202, 2009.

[19] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware standby-sparing technique for periodic real-time applications," in *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pp. 190–197, Oct 2011.

[20] M. A. Haque, H. Aydin, and D. Zhu, "Energy management of standby-sparing systems for fixed-priority real-time workloads," in *2013 International Green Computing Conference Proceedings*, pp. 1–10, June 2013.

[21] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proceedings. 41st Design Automation Conference*, pp. 275–280, July 2004.

[22] N. Fisher, J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 131–140, April 2009.

[23] S. Safari, S. Hessabi, and G. Ershadi, "LESS-MICS: A Low Energy Standby-Sparing Scheme for Mixed-Criticality Systems," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020, doi: 10.1109/TCAD.2020.2977063.

[24] Z. Shirmohammadi, M. Ansari, S. K. Abharian, S. Safari and S. G. Miremadi, "PAM: A Packet Manipulation Mechanism for Mitigating Crosstalk Faults in NoCs," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, Liverpool, 2015, pp. 1895-1902.

[25] S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Energy- Budget-Aware Reliability Management in Multi-Core Embedded Systems with Hybrid Energy Source," The *CSI Journal on Computer Science and Engineering* (JCSE), vol. 15, no. 2, pp. 31-43, 2018.

[26] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2015.

[27] M. Ansari, S. Safari, F. R. Poursafaei, M. Salehi, and A. Ejlali, "AdDQ: Low-Energy Hardware Replication for Real-Time Systems through Adaptive Dual Queue Scheduling," in *The CSI Journal on Computer Science and Engineering (JCSE)*, vol. 15, no. 1, pp. 31-38, 2017.

[28] J. Saber-Latibari, M. Ansari, P. Gohari-Nazari, S. Yari-Karin, A. M. H. Monazzah and A. Ejlali, "READY: Reliability-and Deadline-Aware Power-Budgeting for Heterogeneous Multi-Core Systems," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020, doi: 10.1109/TCAD.2020.3003288.

[29] M. Ansari, J. Saber-Latibari, M. Pasandideh, and A. Ejlali, "Simultaneous Management of Peak-Power and Reliability in Heterogeneous Multicore Embedded Systems," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 623-633, 1 March 2020.

[30] M. Ansari, S. Safari, A. Yeganeh-Khaksar, M. Salehi, and A. Ejlali, "Peak Power Management to Meet Thermal Design Power in Fault-

Tolerant Embedded Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 161-173, 2019.

[31] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, Austin, TX, USA, pp. 3-14, 2001.

[32] M. Ansari, A. Yeganeh-Khaksar, S. Safari, and A. Ejlali, "Peak-Power-Aware Energy Management for Periodic Real-Time Applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 4, pp. 779-788, April 2020.

[33] S. Safari, M. Ansari, G. Ershadi, and S. Hessabi, "On the Scheduling of Energy-Aware Fault-Tolerant Mixed-Criticality Multicore Systems with Service Guarantee Exploration," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2338-2354, 1 Oct. 2019.

[34] Peter Brucker, "Scheduling Algorithms", Springer, fifth edition.

[35] C.L. Liu and James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment," Journal of the Association of Computing Machinery, Vol. 20, No. 1, January 1973, pp. 46-61.

[36] S. Boyd and L. Vandenberghe, "Convex Optimization", 2004.

[37] A. Meixner, M. E. Bauer and D. Sorin, "Argus: Low-Cost, Comprehensive Error Detection in Simple Cores," *IEEE/ACM Int'l Symp. on Microarchitecture (MICRO)*, 2007.

[38] N. George, C. R. Elks, B. W. Johnson, and J. Lach, "Transient fault models and AVF estimation revisited," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, 2010.

[39] S. Yari-Karin, A. Sahraee, J. Saber-Latibari, M. Ansari, N. Rohbani, and A. Ejlali, "A Comparative Study of Joint Power and Reliability Management Techniques in Multicore Embedded Systems," in *2020 CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, pp. 1-8, 2020.

[40] D. Zhu, R. Melhem, and Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Proceedings of Int'l Conf. Computer Aided Design*, pp. 35–40, 2004.

**Amir Yeganeh-Khaksar** received his M.Sc. degree in computer engineering from Sharif University of Technology (SUT), Tehran, Iran, in 2019, and the B.Sc. degree from Ferdowsi University of Mashhad (FUM), Mashhad, Iran, in 2016. From 2016 to 2019, he was a member of Embedded Systems Research Laboratory (ESRLab) at the department of computer engineering, Sharif University of Technology. He was honored to be a member of the national elites foundation in 2019. His current research interests include low power design, real-time embedded systems, and fault-tolerant embedded systems.

**Mohsen Ansari** received the M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2016. He is currently working toward the PhD degree in computer engineering at Sharif University, Tehran, Iran, from 2016 until now. He is now a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany. Also, he is a member of Embedded Systems Research Laboratory (ESRLAB) at the department of computer engineering, Sharif University of Technology. His research interests include low-power design of embedded systems and multi-/many-core systems with a focus on dependability/reliability.

**Alireza Ejlali** received the PhD degree in computer engineering from Sharif University of Technology in, Tehran, Iran, in 2006. He is currently an associate professor of computer engineering at Sharif University of Technology. From 2005 to 2006, he was a visiting researcher in the Electronic Systems Design Group, University of Southampton, Southampton, United Kingdom. In 2006, he joined Sharif University of Technology as a faculty member in the department of computer engineering and from 2011 to 2015 he was the director of Computer Architecture Group in this department. His research interests include low power design, real-time embedded systems, and fault-tolerant embedded systems.