Passive Primary/Backup-based Scheduling for Simultaneous Power and Reliability Management on Heterogeneous Embedded Systems

Sina Yari-Karin, Roozbeh Siyadatzadeh, Mohsen Ansari, and Alireza Ejlali

Abstract—In addition to meeting the real-time constraint, power/energy efficiency and high reliability are two vital objectives for real-time embedded systems. Recently, heterogeneous multicore systems have been considered an appropriate solution for achieving joint power/energy efficiency and high reliability. However, power/energy and reliability are two conflict requirements due to the inherent redundancy of fault-tolerance techniques. Also, because of the heterogeneity of the system, the execution of the tasks, especially real-time tasks, in the heterogeneous system is more complicated than the homogeneous system. The proposed method in this paper employs a passive primary/backup technique to preserve the reliability requirement of the system at a satisfactory level and reduces power/energy consumption in heterogeneous multicore systems by considering real-time and peak power constraints. The proposed method attempts to map the primary and backup tasks in a mixed manner to benefit from the execution of the tasks in different core types and schedules the backup tasks after finishing the primary tasks to remove the overlap between the execution of the primary and backup tasks. Compared to the existing state-of-the-art methods, experimental results demonstrate our proposed method's power efficiency and effectiveness in terms of schedulability.

Index Terms—Embedded Systems, Fault-Tolerance, Real-Time Scheduling, Heterogeneous Multicore Systems, Power/Energy Consumption, Thermal Design Power.

1 Introduction

Technology scaling and high integration of transistors in a single chip have led to utilize of multicore systems, especially in embedded systems [1][2]. In this regard, High performance can be achieved at a low cost through multicore systems [3]. However, due to increased power density, the simultaneous exploitation of all the devices integrated into the chip is impossible [2][3]. In addition, embedded systems consider power/energy consumption as one of the most critical constraints [4][5]. On the other hand, the tradeoff between battery life and energy consumption is an outstanding challenge, especially in energy-constrained real-time embedded systems [1]. Therefore, power/energy efficiency is one of the essential design scopes for embedded systems [5]. Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are two well-known power/energy management techniques that have been studied for several decades. The DVFS technique reduces the power/energy consumption by switching the Voltage/frequency (V/f) levels [6][7]. Yet, in real-time embedded systems, the (V/f) levels are selected such that real-time and reliability constraints are guaranteed [7]. Also, the DPM technique keeps the cores in a low power state by turning off the idle cores [6]. On the other hand, considering peak power constraint such as Thermal Design Power (TDP) can aid in reducing the power/energy consumption and temperature [5].

Besides the power/energy consumption issue, reliability is a life-threatening requirement in real-time embedded systems due to their safety-critical applications [9][11]. In this regard, fault-tolerance techniques are exploited to achieve high reliability [1][6]. Reliability is subjected to different types of faults: (*i*) transient faults, (*ii*) permanent faults [1][5][6]. Transient faults are the most common fault type and are short-lived [6][10]. Typically, task re-execution can unravel transient fault [6][10]. However, permanent faults damage the system components and require hardware redundancy to be tolerated [6][10]. The Standby-Sparing (SS) and Primary/Backup (PB) techniques are the common techniques to deal with both transient and permanent faults [6][10][13]. Multicore systems have an intrinsic redundancy that provides opportunities to implement different SS and PB techniques [6]. The two well-known SS techniques are Hot Standby-Sparing (HSS) and Cold Standby-Sparing (CSS) [8][10][11]. Also, Passive Primary/Backup (PPB) and Active Primary/Backup (APB) are prominent primary/backup techniques [13][26]. The SS technique dedicates one core for the execution of the backup tasks (called spare core) and one for the execution of primary tasks. In contrast, the PB technique utilizes all the cores to execute primary and backup tasks [13][26]. In the HSS and APB techniques, both primary and backup tasks are activated simultaneously [8][10][13][26]. However, in the CSS and PPB techniques, the backup task is idle during the execution of the primary task and is activated

• *S. Yari-Karin, R. Siyadatzadeh, M. Ansari, and A. Ejlali are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. E-mails: {sinayari, siyadatzadeh}ce.sharif.edu, and {ansari, ejlali}@sharif.edu.*

Manuscript received D. M. Y; revised D. M. Y; accepted D. M. Y. Date of publication D. M. Y; date of current version D. M. Y.
(Corresponding author: Alireza Ejlali and Mohsen Ansari)
Recommended for acceptance by X. X.
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.X.X

when a fault occurs on the corresponding primary task [8][10][13][26]. In the HSS and APB techniques, the execution of the backup tasks may have overlapped with the execution of the primary tasks. However, in the CSS and PPB techniques, the overlap between the primary and backup tasks is barred [8][10][11]. SS and PB techniques are active fault-tolerance techniques because they exploit fault detection mechanisms [11], unlike popular approaches such as TMR (N-Modular Redundancy with N = 3), which employs a fault-masking mechanism [10].

For low power/energy system design, since in the realistic-faulty execution scenarios, the fault occurrence is rare, it is preferable to avoid as far as possible the simultaneous execution of the primary and backup tasks to save the power/energy consumption [1][13]. For this reason, the CSS and PPB techniques can be more suitable. However, in real-time embedded systems, meeting deadlines is critical. Thus, exploiting the CSS and PPB techniques may not be applicable [10]. Therefore, the CSS and PPB techniques should be exploited intelligently. Otherwise, they impose extra time overhead [10]. It should be noted that fault-tolerance techniques are mainly based on redundancy that escalates the power/energy consumption of the system such that it makes severe problems in terms of peak power or thermal violation [8]. Therefore, simultaneous management of power/energy and reliability is required. For this purpose, utilizing the heterogeneous platform is an appropriate solution for achieving joint power/energy efficiency and high reliability while meeting the real-time constraint [12][13]. Intuitively, the simultaneous use of High-Performance (HP) and Low-Power (LP) cores has attracted much attention [13]. Fig. 1 depicts the behavior of the different fault-tolerance techniques in a heterogeneous platform[1]. We should mention that each parameter (i.e., peak/average power, energy, and execution time) is normalized separately. Also, due to the heterogeneity of the platform, we have considered two different approaches for the TMR technique. In the TMR-HP approach, two copies of a task are executed in the HP island, and in the TMR-LP approach, two copies are executed in the LP island. It can be seen that the CSS (PPB) technique can be a suitable technique for simultaneous management of power/energy consumption and reliability. Still, it has more time overhead than the HSS (APB), TMR-LP, and TMR-HP techniques. Ergo, simultaneously considering the reliability and real-time constraints while reducing power/energy consumption is a significant challenge [12]. On the other hand, research studies indicate that faults are short-lived and rare events [13]. Therefore, it is good that realistic faulty execution scenarios be considered in the proposed method in order to reduce power/energy further and preserve reliability while considering real-time constraint.

This paper demonstrates that the previously proposed schemes do not appropriately utilize the core type for standby-sparing and primary/backup techniques, especially in realistic faulty execution scenarios. Moreover, the timing overlap and simultaneous execution of primary and backup tasks negatively affect power/energy

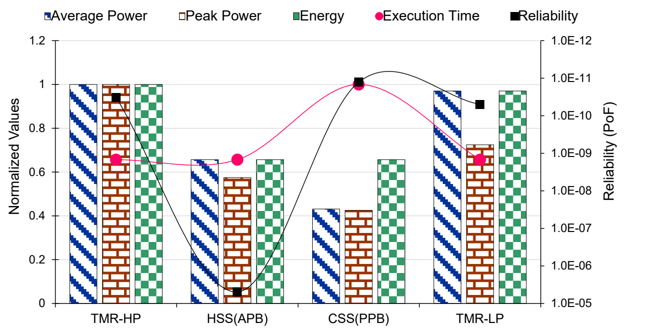

Fig. 1. Comparison of different fault-tolerance techniques on a heterogeneous system.

consumption, especially in the realistic-faulty execution scenarios. Also, the previously proposed methods do not provide a proper simultaneous power/energy and reliability management with an eye toward realistic faulty execution scenarios. Ergo, our proposed method utilizes a heterogeneous system and simultaneous execution of primary and backup tasks in HP and LP islands by the PPB technique to provide appropriate joint power and reliability management, especially for realistic scenarios. Our proposed method attempts to map and schedule the backup tasks after finishing the primary tasks to remove the overlaps between the execution of the primary and backup tasks. In this case, when the primary task completes successfully, its corresponding backup task is dropped, resulting in a significant amount of power/energy saving. Also, we employ the DVFS and DPM techniques in order to reduce power/energy consumption further.

Paper organization: The rest of this paper is organized as follows. We review the related work in section 2. After that, section 3 presents our system model, and section 4 defines the problem. Then, we give the details of our proposed method in section 5. To evaluate the effectiveness of our proposed method, we compared it with state-of-the-art methods, and experimental results are shown in section 6. Finally, we conclude the paper in section 7.

2 Related Work

Performance balancing among power/energy, reliability, and real-time has been discussed for several decades. For instance, Huang et al. [14] proposed a scheduling algorithm to make a tradeoff between energy, reliability, and makespan. To this end, the proposed method in [14] manages the mentioned parameters by finding suitable node allocation and selecting the appropriate frequency level for the execution of the nodes. Also, Hu et al. [15] have proposed a heuristic-based energy management method that reduces energy consumption through processor merging and DVFS techniques while meeting the reliability requirement. On the other hand, the proposed method in [16] guarantees the reliability requirement of the system through the task replication technique (i.e., passive redundancy) and adds enough redundant modules to increase the reliability with respect to minimizing the number of resources and decrease the time complexity. Also, the

[1] The reliability is shown as Probability of Failure (PoF), which is calculated by: PoF = 1- Reliability.

proposed methods in [17] and [18] provide real-time task scheduling for energy and power management, respectively. Yet, these proposed methods do not consider reliability in the proposed methods. Similarly, Yuan et al. [19] have focused on the energy management of distributed cloud systems while preserving the response time limits. In this regard, a simulated-annealing-based algorithm is employed to minimize the energy while considering the performance of servers. Also, the proposed method in [20] provided a task mapping and scheduling for multi-cloud systems to optimize the makespan and total cost while preserving the security and reliability constraints. Furthermore, Kang et al. [21] have presented a scheduling algorithm for a real-time control system that makes a tradeoff between power supply and workload. Still, the proposed method in [21] does not use fault-tolerance techniques. However, this section discusses some methods that employed the Standby-sparing or primary/backup techniques, emphasizing power/energy consumption and reliability management concerning the real-time constraint. Also, Table 1 depicts an overview of related works.

Niu et al. in [1] and [4] have proposed different scheduling algorithms for weakly hard real-time embedded systems to improve the Quality of Service (QoS) and feasibility of the system while considering the reliability and energy constraints. The first scheduling algorithm specifies obligatory jobs that should be executed according to the SS technique under energy budget constraints. However, it has been stated in [1] and [4] that this floating redundant scheme may disturb the schedulability of the task set. For this purpose, the window transferring scheme is proposed, and in the following, an integrated algorithm is presented to improve the performance. Ansari et al. [5] have exploited the SS technique for periodic real-time tasks to keep peak power under TDP constraint. To this end, the proposed method separates the execution policy of the primary and backup cores. The primary and backup cores execute the tasks according to the PPA-EDF and PPA-EDL policies, respectively, to minimize the overlap between the execution of primary and backup tasks. Roy et al. [6] have used the SS technique on a heterogeneous multicore system that included high-performance and low-power cores. The proposed method in [6] finds the appropriate core type to execute the primary and backup tasks. It has also been stated that consciously selecting the core type and amount of overlap between the execution of the primary and backup tasks significantly reduces energy consumption.

The proposed three-layer framework in [8] consists of a feedback system in the primary cores. The proposed method spreads the slack time by a heuristic approach to minimize overall energy consumption. Also, the real-time constraint is considered in the feedback system. Roy et al. [13] have proposed a framework to reduce energy consumption while tolerating faults. The Reverse Preference-Oriented Priority Assignment (RPPA) proposed method has two offline and online phases. The RPPA method exploits a dual queue-based technique to delay the execution of the backup tasks as much as possible, and the DVFS technique for primary tasks, which will be effective for energy saving. Ejlali et al. [22] have proposed a hardware redundancy technique for dual-core real-time systems that executes dependent frame-based tasks. In this work, the primary core uses DVS, and the backup core uses DPM. Haque et al. have proposed the SSFP algorithm in [23] and attempt to execute the backup tasks as late as possible by using a dual-queue mechanism to reduce energy consumption. Ejlali et al. have proposed a low-energy standby-sparing in [24]. The proposed method in [24] is an online energy management technique for a low-energy standby-sparing (LESS) system, which considers voltage transition and activation overheads forced by DVS and DPM. This method is used in a dual-core hard real-time system and uses dynamic slack times to reduce energy consumption while guaranteeing hard deadlines.

Table 1. Overview of the related work

Ref. #	Task Model	Techniques		
		Fault-Tolerance	Power Constraint	Power Management
[1],[4]†	Periodic	HSS	×	DVS, DPM
[5]†	Periodic	HSS	✓	DVFS, DPM
[6]‡	Frame-based	HSS	×	DVFS, DPM
[8]†	Task Graph	HSS	×	DVS, DPM
[13]‡	Periodic	APB	×	DVFS, DPM
[22]†	Task Graph	HSS	×	DVS, DPM
[23]†	Periodic	HSS	×	DVS, DPM
[24]†	Frame-based	SS	×	DVS, DPM
[25]†	Periodic	HSS	×	DVS, DPM
[26]†	Periodic	CSS	×	×
[27]†	Periodic	APB	×	DVFS, DPM
[28]†	Periodic	HSS	×	DVS, DPM
[29]‡	Periodic	APB	×	DVFS, DPM
[30]†	Frame-based	APB	✓	DPM
[31]‡	Frame-based	CSS	×	DPM, DFS
[32]‡	Task Graph	CSS	✓	DPM, DVFS
Proposed Method‡	Periodic	PPB	✓	DVFS, DPM

System Model: †→ Homogeneous ‡→ Heterogeneous

Haque et al. [25] employed EDF scheduling on the primary core and EDL scheduling on the backup core for periodic real-time tasks to execute primary tasks as soon as possible and delay the execution of backup tasks. The proposed method in [26] reduces the rejection of the critical jobs by employing a CSS mechanism. Purushothaman et al. [26] employed the CSS technique, and the Fault-Tolerant Fair Scheduler (FT-FS) proposed method executes tasks fairly by considering the utilization and distributes the slack times of jobs to minimize the job terminations/rejections. Guo et al. [27] have proposed the Paired-Standby-Sparing (Paired-SS) and Generalized-Standby-Sparing (Generalized-SS) methods. In the Paired-SS method, cores are divided into pairs, and the standby-sparing method applies to pairs. On the other hand, the Generalized-SS divides the cores into primary cores and secondary cores. The primary tasks are executed in the primary cores under partitioned-EDF, and the DVFS technique and backup tasks are executed in secondary cores under the partitioned-EDL and DPM mechanism. Zhang [28] has proposed the EAMPSA scheduling algorithm to reduce the energy of real-time task sets. The EAMPSA proposed method exploits EDF/DDM scheduling algorithms to access the shared resources. Also, the mixed mapping partitioning policy is employed to select the appropriate core type and

V/f level for the execution of the primary and backup tasks and reduce the energy consumption.

Roy et al. [29] have proposed an SS-based method that exploits task partitioning and assigns the frequency of tasks at run time to hold energy consumption at a minimum level and meet fault-tolerance constraints for real-time tasks on heterogeneous multicore systems. Ansari et al. [30] have provided a power-aware scheduling method for framed-based task sets. The proposed method considers the power profile of the task set and maps the primary and backup tasks to the paired cores with minimum utilization. After that, the proposed method schedules the primary and backup tasks based on Maximum-Peak-Power-First (MPPF) and Maximum-Peak-Power-Last (MPPL) policies, respectively, to reduce the overlap between the execution of primary and backup tasks. Also, the proposed method keeps the power consumption below the TDP constraint to reduce power consumption further. The FEST proposed method in [31] provides energy-aware fault-tolerant scheduling on a heterogeneous system by employing the CSS technique and backup-backup overloading mechanism. For this purpose, the tasks are selected in nonincreasing order of their execution times, and there is an overloading window for the execution of the backup task to reduce the energy consumption. The TASS proposed method in [32] provides a thermal-aware power management method that executes the task set under Thermal Safe Power (TSP) constraint to improve the QoS. In this regard, the proposed method balances the utilization of the task set by considering a core pair for the execution of the primary and backup tasks.

Summary: From the discussed methods, it can be seen that none of the existing active fault-tolerance-based methods had reduced the power/energy of a heterogeneous system by applying a PPB technique and considering a peak power constraint. To the best of our knowledge, our proposed method is the first work that employs the PPB technique to reduce power/energy consumption while meeting the peak power consumption and real-time constraints.

3 Background

In this section, we explain the assumptions made for the task, power, platform, and reliability models.

3.1 Task Model

The task model which will be used in our proposed method corresponds to tasks with implicit deadlines. The task set executed by our proposed method includes n periodic tasks $\varpi = \{T_1, T_2, \ldots, T_n\}$. Each task T_i has a worst-case execution time ($WCET_i$), a deadline d_i, and a period p_i [5][7][26]. Also, the task T_i will be executed under DVFS condition and at V_c/f_c level. Similarly, the hyperperiod of the task set is equal to the least common multiple among the period of all tasks ($H = LCM\ (\forall\ p_i)$) [5][7][26]. The j^{th} job of task T_i releases at the time $(j\text{-}1)\times p_i$, and the deadline for this job j is equal to $j\times p_i$ [7][12][33]. Also, the system utilization (U_{sys}) can be obtained by $\sum \frac{WCET_i}{p_i}$ [5][7][26]. Similarly, we consider a backup task B_i for each task T_i and denote the j^{th} job of B_i by JB_{ij}.

3.2 Power and Energy Model

We study a heterogeneous multicore system that consists of M_{HP} number of HP cores, which are power-hungry, and M_{LP} number of LP cores that are power-efficient [6][34]. In this paper, we consider the system consists of m cores, where $m = M_{HP}+M_{LP}$. Also, the power consumption consists of static and dynamic components and, under DVFS conditions, can be calculated by means of Eq.1 [1][5][7][8]:

$$\begin{aligned} P_{core}(V_c, f_c) &= P_{Static} + P_{Dynamic} \\ &= \rho_c\left(I_0 e^{\frac{-V_{th}}{\eta V_T}} V_{max}\right) + \rho_c^3\left(\alpha_i^{HP\_or\_LP} C_L V_{max}^2 f_{max}\right) \end{aligned} \tag{1}$$

Where C_L is the switching capacitance, η is a technology parameter, $\rho_c = \frac{V_c}{V_{max}} = \frac{f_c}{f_{max}}$ is normalized voltage and frequency coefficient under the DVFS condition [33], and $\alpha_i^{HP\_or\_LP}$ is the activity factor for the task T_i. Due to heterogeneity nature of the assumed system, each task exhibits different power consumption characteristics on different types of cores. Therefore, we use the HP and LP superscripts to distinguish between the task power consumption parameter values on the HP and LP core, respectively. Finally, the energy consumption of task T_i can be calculated as Eq. 2 [1][8][33][34]:

$$Energy(T_i) = \rho_c^2\left(\alpha_i^{HP\_or\_LP} C_L V_{max}^2 f_{max} WCET(T_i)\right) \tag{2}$$

3.3 Fault and Reliability Model

In this paper, thanks to the PPB technique, we can tolerate both permanent and transient faults. Transient faults are commonly modeled as a Poisson distribution with a fault rate that is based on a function of the supply voltage changes [34][35]. In our evaluations, the transient fault rate is considered as Eq. 3 [22][34][35]:

$$\lambda(V) = \lambda_0 \times 10^{\frac{V_{max}-V}{d}} \tag{3}$$

Where $\lambda_0 = 10^{-6}$ is the transient fault rate at V_{max} and d specifies the sensitivity of the system to voltage scaling [5][33][34]. The functional reliability of a task can be written as Eq. 4 [5][34][35]:

$$R(T_i) = e^{-\lambda(V_i)\times FVI\times WCET_i} \tag{4}$$

Where FVI is the Function Vulnerability Index [34]. The reliability of the primary and backup tasks is independent. Ergo, the reliability of the PPB technique for each task is similar to Eq. 5 [10]:

$$R(T_i, TB_i) = R(T_i) + (1 - R(T_i)) \times R(TB_i) \tag{5}$$

Correspondingly, the reliability of a system with n tasks executed can be calculated as Eq. 6 [5][10][34]:

$$R_{system} = \prod_{i=1}^{n} R(T_i, TB_i) \tag{6}$$

Safety-critical applications consider a reliability target to guarantee the correct execution of the tasks. For this purpose, the *Number of Nines* is defined, which depicts an agreed level of reliability, and presents the availability of the system [10][34]. For example, DO-178B is an avionic standard that describes five reliability levels from A (the lowest level) to E (the highest level) [36]. The Number of Nines value can be calculated as Eq. 7 [36][37]:

$$Number\ Of\ Nines = \lfloor -log_{10}(1 - R_{system}) \rfloor \tag{7}$$

This paper uses the processor pair technique and considers a pair (C_{pair}= {$C_{primary}$, C_{backup}}) to execute each task [2][10]. The process pair technique executes two identical versions of tasks in different cores, and it reduces hardware communications that are proper for applications necessitating high availability [10]. It has been shown in the Argus mechanism that checking of control flow, computation, data flow, and memory invariants at run time is sufficient for detecting all possible single errors [38]. Therefore, if no fault has been detected during the execution of the first copy of the tasks, their results will be supplied to the system, and the second copy of the tasks will be dropped. In this regard, we implement the fault detection mechanism similar to the [1] and [24] at the software level, and the faults can be detected at the end of the execution of the jobs by sanity check (or consistency check). Consequently, our proposed method can guarantee the tolerance of transient and permanent faults as follows:

- Transient faults can be tolerated during the execution of primary task instances.
- A permanent fault (i.e., malfunctioning of a processing component) is guaranteed to be tolerated during the absence of transient faults in the execution of the task set.

Similarly, the time overhead of the fault detection and setup time of the PPB technique is integrated within the worst-case execution time of the tasks [2][24].

4 Problem Statement

We define the problem of this paper in the following mathematical notation. It should be noted that this paper aims to minimize the total power consumption with a series of constraints. The problem is how to find the task-to-core assignment, the scheduling of the tasks (find the finish time (e_{ij}) and the start time (s_{ij}) of all primary and backup jobs), and the V/f level for each job of tasks to achieve power efficiency. In this mathematical formulation, n is the number of tasks, v is the number of available Voltage/frequency levels for each core, and m determines the total number of the cores. The task-to-core mapping and V/f level assignment are demonstrated by the matrix $X \in \{0,1\}^{n\times m\times v}$. The task i is mapped to the core k and is executed under the *V/f* level l if and only if $X_{ikl} = 1$. The goal of our proposed method is to minimize the power consumption of the system such that the real-time and peak power constraints are met. Table 2 summarizes the notations, and we formulate the stated problem in the following.

Problem: Minimize the total power consumption by taking the peak power and real-time constraints to account for and guarantee reliability at an acceptable level. The power consumption of the j^{th} job of task i is obtained by Eq. 1. Consequently, the total power consumed by the system is equal to the sum of the power consumed by all the jobs performed in a hyper period.

$$Minimize\ P_{total} = \sum_{i=1}^{i=N} \sum_{j=1}^{j=n_i} P_{ij}(V/f) \tag{8}$$

Task Timing Constraint: The execution of the j^{th} job (primary and backup jobs) of task i at the frequency f_{ij} should

Table 2. A few important Notations

ϖ	Task set	M	Number of cores
T_i	i^{th} task	$Energy(T_i)$	Energy of task i
TB_i	i^{th} backup task	$P(T_i)$	Power of task i
J_{ij}	j^{th} job of task i	H	Hyperperiod
p_i	Period of task i	$R(T_i)$	Reliability of task i
U_{sys}	Utilization of system	$X\in\{0,1\}^{n\times m\times v}$	Mapping matrix
R_{system}	Reliability of system	V/f	Voltage/frequency level
M_{LP}	Number of cores in Low-power Island		
M_{HP}	Number of cores in High-performance Island		
JB_{ij}	j^{th} backup job of backup task i		
d_{ij}	Deadline of j^{th} job of task i		

be completed before the job deadline.

$$\forall t_i \in \varpi \& \forall j \in t_i: j \times p_i \le (j-1) \times p_i + WCET(f_{ij}) \le d_{ij} \tag{9}$$

Task to Core and V/f level Assignment Constraint: each task will be assigned to one and only one primary/backup core and will be executed under a single V/f level.

$$\forall i,k,l: \sum X_{ikl} \le 1 \tag{10}$$

Task Priority Constraint: The primary tasks are assigned to high priority, and backup tasks are in low priority.

Chip Frequency Constraint: Reducing the frequency level to below the energy-efficient frequency level f_{ee} leads to a negative effect on energy consumption [6]. Therefore, the cores frequency should have a value between f_{ee} and f_{max}.

$$f(core_i) = \rho_i f_{max} \ :\quad f_{ee} \le \rho_i f_{max} \le f_{max} \tag{11}$$

Utilization Bound of the System and the Cores: The sum of all task utilization on the system should be less than the number of cores, and the sum of all task utilization mapped on each underlying core k should be less than 1.

$$\forall\ i,k,l: \sum_{k} X_{il} U_k \le 1 \tag{12}$$

Solving the above problem with the mentioned constraints and finding proper scheduling for a multicore system to minimize power consumption exponentially increases when the size of the problem increases at run-time [2][5]. Therefore, the problem is considered an NP-hard problem [2][5]. With the aid of a heuristic, our proposed method provides a proper solution for power efficiency such that it meets peak power, reliability, and real-time constraints.

5 Proposed Method in Detail

We turn our attention to the task mapping/scheduling problem with the given constraints and requirements in this section. It should be noted that due to the complexity of the problem, providing an efficient optimal solution is not applicable due to the need for a large run-time for optimal solutions [39]. For this purpose, heuristic-based methods are common in practice [39]. Our heuristic-based proposed method has two offline and online parts. Fig. 2 provides the procedure of our proposed method. In the offline part, the appropriate core type and V/f level for the execution of the primary and backup tasks will be determined. Afterward, the scheduling policy is applied such that peak power and real-time constraints are preserved. In the online part, if a fault is detected in the execution of the primary task, the backup task starts the execution.

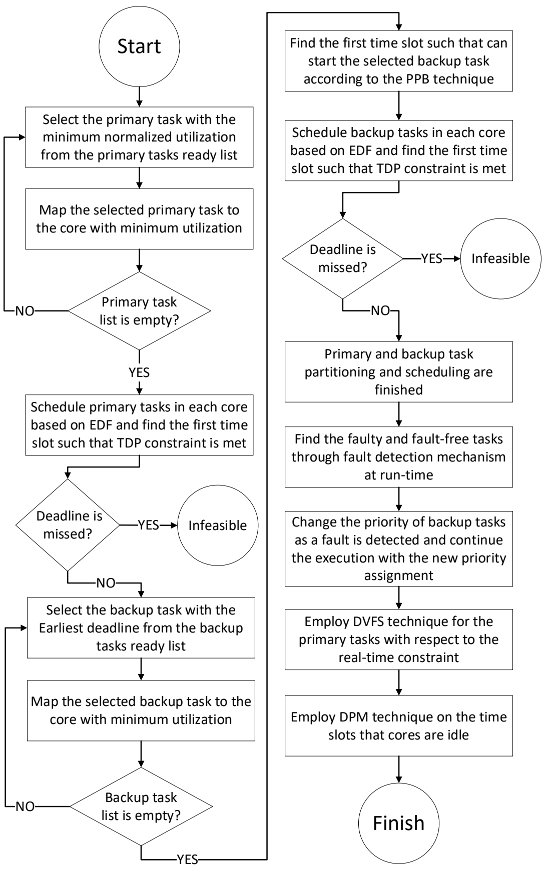
Start
Select the primary task with the minimum normalized utilization from the primary tasks ready list
Map the selected primary task to the core with minimum utilization
Primary task list is empty?
NO
YES
Schedule primary tasks in each core based on EDF and find the first time slot such that TDP constraint is met
Deadline is missed?
YES
Infeasible
NO
Select the backup task with the Earliest deadline from the backup tasks ready list
Map the selected backup task to the core with minimum utilization
Backup task list is empty?
NO
YES
Find the first time slot such that can start the selected backup task according to the PPB technique
Schedule backup tasks in each core based on EDF and find the first time slot such that TDP constraint is met
Deadline is missed?
YES
Infeasible
NO
Primary and backup task partitioning and scheduling are finished
Find the faulty and fault-free tasks through fault detection mechanism at run-time
Change the priority of backup tasks as a fault is detected and continue the execution with the new priority assignment
Employ DVFS technique for the primary tasks with respect to the real-time constraint
Employ DPM technique on the time slots that cores are idle
Finish

Fig 2. Flow diagram of the proposed method.

Otherwise, the execution of the backup task will be prevented to reduce power consumption. In the following, the proposed policies for each part will be discussed.

5.1 Partitioning/Allocation and Scheduling

The purpose of task partitioning and allocation is to provide an appropriate task mapping such that reliability, real-time, and peak power constraints are met. For this purpose, the PPB technique, which is exploited in our proposed method (see subsection 3.3), improves the reliability of the system by considering the sum of two probabilities in the execution of primary and backup tasks: (*i*) The probability that the primary task operates correctly ($R(T_i)$). (*ii*) The probability that the backup task operates correctly while the primary task has failed and has been replaced by the backup task ($(1 - R(T_i)) \times R(TB_i)$). However, there are various mechanisms for task partitioning/allocation. The core pair mechanism divides the cores into the primary and backup cores. In this mechanism, each type of task (i.e., primary/backup) is assigned to a core of the same type (primary/backup) as [1] and [5]. In addition, to equilibrate the workload of the cores, tasks can be mapped to the cores with the minimum utilization first policy. Fig. 3a depicts the pair core mechanism in a homogeneous system. It can be seen that primary/backup tasks are mapped to the primary/backup cores according to their utilization. Yet, this mechanism is not necessarily an appropriate task mapping solution due to poor resource allocation.

Task mapping/allocation in heterogeneous systems is

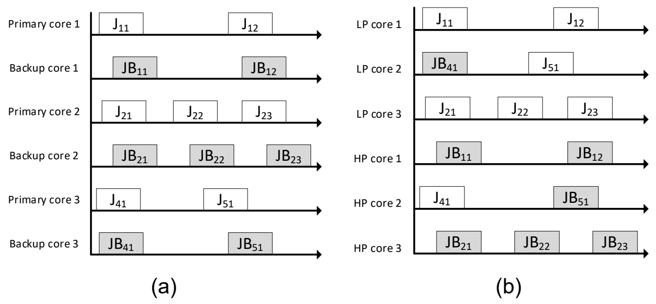

Fig 3. Task mapping mechanisms, a) Core pair on a homogeneous system, and b) alternative selection on a heterogeneous system.

more complicated than in homogeneous systems due to the different characteristics of tasks on each island. It should be mentioned that in the heterogeneous setting, different core types are combined in the same chip. Heterogenous architectures mostly include two low-power and high-performance islands [13]. The cores on high-performance island execute tasks with less execution time and more reliability [13][34]. On the other hand, the cores on low-power island execute tasks with more execution time and less power consumption [13][34]. Therefore, a heterogeneous multicore system can make a tradeoff between power/energy efficacy and high reliability and performance. Fig. 3b depicts the alternative selection mechanism proposed in [13]. In this proposed method, the primary tasks are first mapped to the proper cores according to their utilization. Then, the backup tasks will be mapped to the respective alternative core. For example, the mapping/allocation mechanism decides to map the primary task T_4 to the HP core 2, and the alternative backup task for this core is LP core 2; therefore, the backup task (i.e., TB_4) will be mapped to this core. However, our proposed method employs a different mechanism for mapping/allocation to map the primary and backup tasks efficiently to be able to employ the simultaneous execution of tasks on both HP and LP islands to reduce power/energy consumption while meeting the real-time constraint.

Our mapping/allocation mechanism considers different policies for selecting the appropriate core type for executing the primary and backup tasks to be power efficient. We consider two priority lists for making the ready task lists. The primary tasks are assigned to the high priority list, and the backup tasks are stored in the low priority list. The purpose is to eliminate the overlap between the execution of primary and backup tasks. By prioritizing the primary and backup tasks, low-priority tasks can be executed as late as possible. Therefore, the overlap between the execution of the primary and backup tasks can be avoided. Consequently, power consumption will be reduced. After prioritizing the tasks by generating the ready lists, it is time to map/allocate the primary and backup tasks. For this purpose, the tasks with high priority (i.e., primary tasks) are mapped to the cores according to their utilization. However, due to the heterogeneity of the system, the execution times of the tasks can be different in LP and HP cores. Thus, we use the nominal utilization of the primary tasks for mapping, which is calculated by Eq. 13 [13]:

$$U_i^{nom} = \frac{WCET_i^{HP}(f_{max}^{HP})}{P_i} \quad (13)$$

The primary tasks will be selected according to the Minimum Nominal Utilization at First (MNUF) policy and are mapped to the core with maximum free capacity (minimum utilization). Therefore, primary tasks are distributed among all the LP and HP cores, and we benefit from the execution of the primary tasks on both islands. When all the primary tasks are mapped to the proper core, we will find the first time slot to execute tasks without real-time and peak power violation. For this purpose, each core exploits peak power-aware EDF, and the feasibility of the task set will be checked during the scheduling of the tasks.

Overlap elimination by considering the real-time and peak power constraints is an essential part of the backup tasks mapping/allocation. For this purpose, we employ a different mechanism for determining proper core type selection for the backup tasks. In this regard, the backup tasks are selected according to the earliest deadline first policy and will be mapped to the core with the minimum utilization concerning the PPB technique. Consequently, out of the set of backup tasks in the low priority list, the task with the earliest deadline is executed as soon as possible. After selecting the appropriate core type, we find the first time slot such that the backup tasks (Considering the PPB technique) simultaneously guarantee the peak power and real-time constraints. Therefore, in order to have a schedulable task set according to our proposed method, which exploits the EDF algorithm, the primary/backup tasks should meet the Eq. 12 requirement during core selection for the execution of the primary/backup tasks. Also, the worst-case response time (WCRT) of each job should consider the worst-case execution of prior high-priority jobs. Thus, the WCRT of task i (primary/backup) should meet the iterative Eq. 14 requirement [40]:

$$WCET_i + \left(\sum_{j=1}^{i-1} \left\lceil \frac{WCRT_i}{P_j} \right\rceil \times WCET_j \right) \leq d_i \qquad (14)$$

5.2 Run-Time Procedure

Our proposed method executes tasks in a priority-based manner. First, the primary jobs (i.e., high priority list) are executed. At the end of the execution of each primary job, the sanity check is performed to detect the occurrence of a fault in the execution of the task. It should be noted that in this paper, we do not get involved in how to design the sanity checker. Ergo, if the execution of the primary job is detected to be faulty, the priority of the corresponding backup job will be changed to high in order to execute as soon as possible. Also, in order to reduce power/energy further, DVFS is employed in the primary jobs, and the DPM technique is employed on the idle cores. The idea of overlap elimination in the execution of the primary/backup task leads to a decrease in the power/energy consumption and resource demand, especially in realistic scenarios [26]. Still, our proposed method reduces the power/energy consumption with an eye to schedulability in normal and pessimistic situations. Because it could negatively affect the schedulability of the system in the worst-case scenarios. Therefore, our proposed method is more suitable for systems where low power consumption and high reliability take precedence over schedulability.

Algorithm 1. Mapping policy

```
Input: ready task set $\varpi$= {T1, T2, …, Tn} with the worst-case execution time on LP and HP islands, periods, deadlines. set of cores C= {C1, C2, …, Cm}
Output: Task mapping
Begin
1: $\varpi_1$←sort($\varpi$. $U_i^{nom}$); //insert primary tasks w.r.t nominal utilization
2: $\varpi_2$←sort($\varpi$, $d_i$);   //insert backup tasks w.r.t their deadline
--------------------------Mapping of the primary tasks----------------------
3:  while ( $\varpi_1$ is not empty) do
4:      $T_i$= $\varpi_1$.remove();          //select the primary task
5:      $T_i$.setPriority("high");
6:      $\varphi$ = C.$min_{utilization}$();   //Find a core with the minimum utilization
7:      $\varphi$.add($T_i$);
8:      update($\varphi$.$U_i$);  //update the utilization of the selected core
9:      $\varpi_1$.remove($T_i$);
--------------------------Mapping of the backup tasks-----------------------
10: while ( $\varpi_2$ is not empty) do
11:     $TB_i$= $\varpi_2$.remove();          //select the backup task
12:     $TB_i$.setPriority("low");
13:     $\varphi b$ = C.$min_{utilization}$();  //Find a core with the minimum utilization
14:     if (! $\varphi b$.contains($T_i$))
15:         $\varphi b$.add($TB_i$);
16:         update($\varphi b$.$U_i$); //update the utilization of the selected core
17:         $\varpi_2$.remove($TB_i$);
End
```

5.3 Algorithm Discussion

Algorithm 1 presents our proposed mapping policy to select the appropriate core type for primary and backup tasks. First, the primary tasks are sorted according to Eq. 13, and the backup of the corresponding primary tasks will be sorted based on the earliest deadline first policy (lines 1-2). After that, we set the priority of the primary tasks to "high" to execute them as soon as possible, and a task with minimum nominal utilization will be selected, and our mapping policy finds the core with minimum utilization for the corresponding primary task. The algorithm iterates until all the tasks are assigned to a core with minimum utilization (lines 3-9). However, the task mapping policy for backup tasks is different (lines 10-17). We set the priority of backup tasks to "low". Then, the selected backup task is mapped to the core with the minimum utilization concerning the PPB technique.

After identifying the appropriate core type for the execution of the primary and backup tasks, algorithm 2 provides our scheduling policy. We attempt to reduce power/energy consumption by decreasing the resource demand. To this end, the offline scheduler activates the backup jobs after the execution of the primary jobs so that no additional resources are needed for power management. First, we calculate the hyperperiod by calculating the least common multiple to get the amount of required time for the execution of all the jobs and assume that the task set is feasible at the beginning (lines 1-2). Then, we calculate the number of jobs for each task (lines 3-4), and the jobs will be added to the selected core from algorithm 1 (lines 5-6). The proposed scheduling policy is applied for the execution of the jobs in lines 7-24. First, we extract the first eligible job in each core based on assigned priority and EDF scheduling policy (lines 8-9). Since the execution

Algorithm 2. Scheduling policy

```
Input: ready task set ϖ= {T1, T2, …, Tn} with the worst-case execution
time on LP and HP islands, periods, deadlines, and power profile
on LP and HP islands, set of cores C= {C1, C2, …, Cm}, available V/f
levels, and TDP.
Output: Task scheduling
----------------------------------Scheduling policy------------------------------
Begin
1:  h=LCM(∀periods);   //Compute the hyperperiod
2:  feasibility= true;
3:  for each task list in Ci of C do
4:      Ci.nk = $\frac{h}{T_i.d}$ ; // calculate the number of jobs of task k
5:      for each job of task nk do
6:          add.Ci (nkj); // add all the jobs of the tasks to the ready list
7:  while (time ≤ h) do
8:      for each core Ci of C do
9:          jik ← extractFirstJob (Ci); // extract the first eligible job
10:             if (jik.isPrimary()) // the type of job is primary
11:                 for each time_slot in jik× pik do
12:                     if (jik.PeakPower + current_power ≤TDP)
13:                         schedule (jik.d, EDF); // schedule jik based on EDF
                                        in the first time slot that TDP is met
14:                         slack← extractSlackTime (jik.d, time_slot);
15:                         vf_DVFS(jik.d); // V/f level w.r.t to the deadline
16:                         backupTime(jik, jBik); // set start time for backup job
17:             else if (jikIsBackup && checkPriority (jBik) && jikIsReady)
18:                 for each time_slot in jik× pik do
19:                     if (jik.PeakPower + current_power ≤TDP)
20:                         schedule (jik.d, EDF); // schedule jik based on EDF
                                        in the first time slot that TDP is met
21:             if (jik_deadline_is_missed()) // check the feasibility
22:                 feasibility= false;
23:                 return feasibility;
24:     time← next_time_slot;
End
--------------------------------------Run-time Events -----------------------------
Event: start execution of the jobs
Event: Employ DVFS
Event: job is completed → remove job
Event: fault detection
            change priority of corresponding backup job to high priority
Event: fault is not detected
            remove the corresponding backup job from the execution list
```

policy is based on priority, we adopt different mechanisms for the execution of primary and backup jobs. For each primary job, based on the EDF scheduling algorithm and observing the TDP peak power constraint, the first available time slot that the real-time constraint is met will be selected (lines 10-13). Then, if a job has available slack time, the V/f level will be determined for the DVFS event to reduce the energy consumption at run-time (lines 14-15). For this purpose, the available slack time of each job is used to calculate the V/f level ratio of the DVFS technique by means of $r(f_c) = \max\{f_{ee}, \frac{WCET(J_{ik})}{WCET(J_{ik})+slack}\}$ [25]. Also, it should be mentioned that we use the DVFS technique for the primary jobs in order to reduce energy, and the backup jobs are executed at the maximum V/f level. Finally, we set the start time of the backup jobs after completing the primary jobs to execute the backup jobs as late as possible in line 16. Backup jobs will also be scheduled according to the real-time and peak power constraints in lines 17 to 20. Still, all the high-priority jobs on the ready list should have been scheduled previously. Also, a feasibility warning signal will be activated when the scheduling cannot meet the real-time constraint (lines 21-23). Our proposed method employs events in the execution of the tasks at run-time. The execution event is responsible for starting the execution of the task set. The DVFS event employs the DVFS technique in the independent time slots that the real-time constraint is guaranteed at run-time. At the end of each job, the completion event sends the signal to the operating system to remove the executed jobs. In the case of fault detection, the fault detection event decides to remove the corresponding backup job or change its priority to high.

6 EXPERIMENTAL EVALUATIONS

In this section, we evaluate the effectiveness of our proposed method in terms of power/energy consumption, reliability, and schedulability.

6.1 Experiment Environment

We have exploited the gem5 full-system simulator [41] to simulate our system model. Also, the McPAT simulator [42] has been employed to obtain the required information, such as power/energy and execution time. We employ a system-level discrete event simulation to evaluate the effectiveness of the proposed method. Also, we consider the transient faults in the evaluations, and the faults are injected by considering a fault vector that determines at which time points faults occur. We have used the MiBench [43] and PARSEC [44] benchmarks to bring the simulations closer to the real world. Fig. 4 depicts the procedure of our simulation setup, and Fig. 5 demonstrates the execution time, power, and reliability of different MiBench and PARSEC programs in both LP and HP islands.

6.2 Compared Approaches and Case Study

In order to show the effectiveness of our proposed method, we compare it with two state-of-the-art methods, i.e., P2AEM [5] and RPPA [13]. (*i*) P2AEM [5]: The P2AEM scheme reduces energy consumption by observing peak power and real-time constraints. In this scheme, the primary tasks are executed according to the PPA-EDF scheduling mechanism. On the other side, backup tasks are executed based on PPA-EDL scheduling in order to minimize overlap between the execution of primary and backup tasks. However, this scheme is proposed for homogeneous

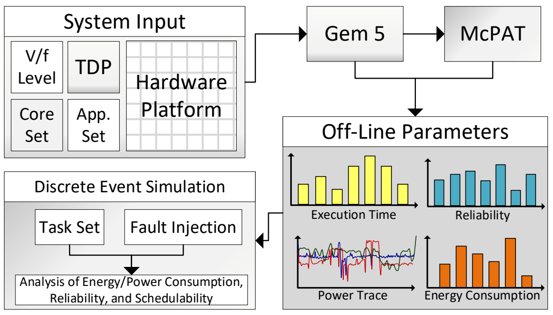

Fig 4. Simulation setup flow.

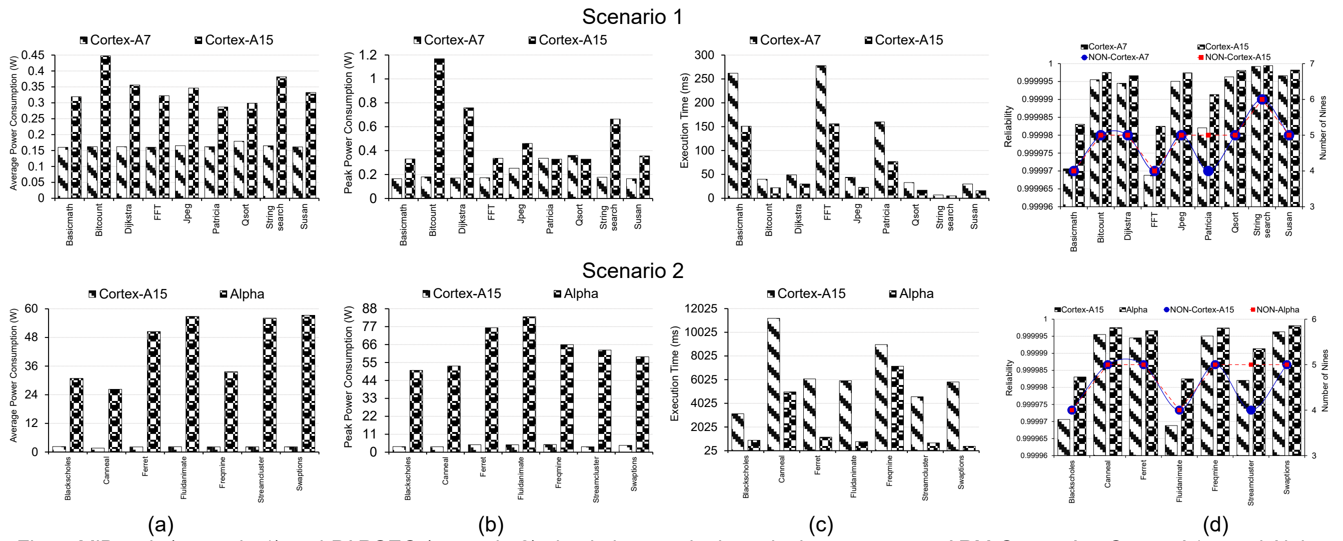

Fig 5. MiBench (scenario 1) and PARSEC (scenario 2) simulation results in a single core system ARM Cortex-A7, Cortex-A15, and Alpha processors, a) Average power, b) Peak power, c) Execution time, and d) Reliability.

systems and uses a core pair mechanism. (*ii*) RPPA [13]: The RPPA scheme maps the primary tasks according to the nominal utilization and selects the core type based on alternative selection to execute backup tasks in a heterogeneous system. Also, the scheduling of the RPPA scheme employs a dual-queue-based task delaying to maximize the backup task cancelation occasions. However, the proposed scheme does not consider peak power consumption. It should be noted that we assume the power/energy efficient core pair scenario for the P2AEM method. In this regard, the primary and backup tasks map to the LP and HP islands, respectively.

To have an accurate evaluation and present the effectiveness of our proposed method compared to the state-of-the-art methods, we have compared the methods in both realistic-case and worst-case scenarios. In the realistic-case scenario, we consider the stochastic nature of faults in the system ($\lambda = 10^{-6}$ faults/µsec) [34]. Therefore, if the primary task is executed correctly, it will be prevented from executing the corresponding backup task. On the other hand, we assume that fault will be detected in executing all the primary jobs in the worst-case scenario. Consequently, all the backup tasks should be executed.

In order to show the effectiveness of our proposed method, we illustrate a case study example. We consider a quad-core heterogeneous system that consists of two LP and two HP cores. Given a periodic task set with four periodic tasks, each task has different worst-case execution times and power consumptions on LP and HP cores due to the heterogeneity of the system. Also, all the tasks arrive at t=0 and have different periods. In this example, for simplicity, we consider the period of the tasks is equal to the deadlines. Therefore, D_1=D_3= 15ms and D_2=D_4= 30ms are the deadlines of tasks. Also, the worst-case execution time of tasks are assumed to be {wc_1^{LP}=5ms, wc_2^{LP}=10ms, wc_3^{LP}=8ms, wc_4^{LP}=7ms} and {wc_1^{HP}=2ms, wc_2^{HP}=7ms, wc_3^{HP}=5ms, wc_4^{HP}=4ms} on LP and HP cores, respectively. We assume the average power consumption is equal to the peak power consumption. Ergo, the power consumption of tasks is considered to be {P_1^{LP}=0.5w, P_2^{LP}=2w, P_3^{LP}=1.5w, P_4^{LP}=1w} and {P_1^{HP}=1w, P_2^{HP}=2.5w, P_3^{HP}=2w, P_4^{HP}=1.5w} on LP and HP cores, respectively.

Fig. 6a depicts the combination of the core pair mechanism along with peak power-aware scheduling. It can be seen that the proposed mechanism in this method always preserves the peak power consumption less than the TDP constraint. However, the lack of an appropriate core type selection in the core pair mechanism leads to real-time constraint violation due to inefficient use of cores capability. On the other hand, as can be seen in Fig. 6b, the alternative core type selection can act better in ensuring the real-time constraint. However, the peak power constraint should be considered to reduce power consumption further. In this regard, our proposed method provides a better task mapping and scheduling, as has been shown in Fig. 6c. Therefore, we simultaneously preserve the peak power

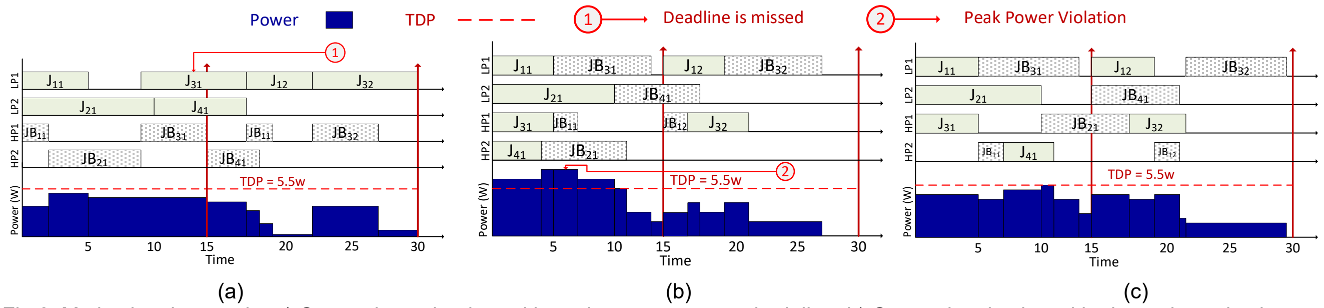

Fig 6. Motivational example, a) Core pair mechanism with peak power-aware scheduling, b) Core pair selection with alternative selection mechanism, and c) Our proposed method.

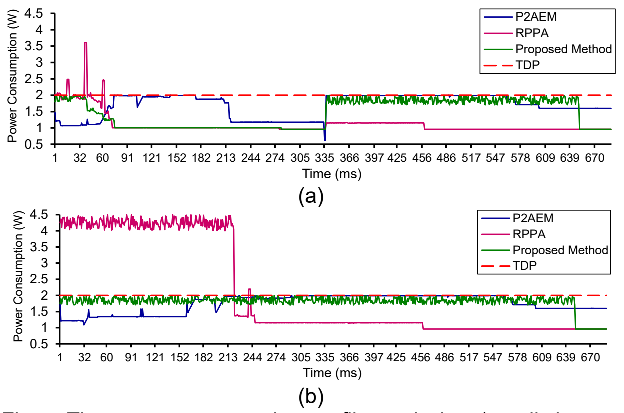

Fig 7. The power consumption profile analysis, a) realistic scenario, and b) worst-case scenario.

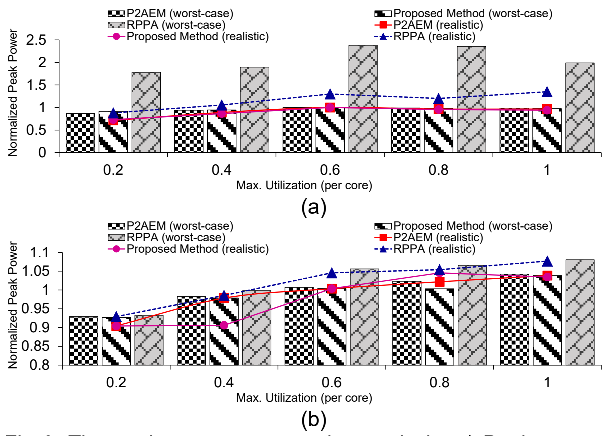

Fig 8. The peak power consumption analysis, a) Peak power in scenario 1, and b) Peak power in scenario 2.

consumption and real-time constraints together with guaranteeing reliability.

6.3 Power/Energy Consumption Analysis

In order to show the impact of workload variation in power/energy consumption, we have considered different utilization in the evaluations. Also, we have randomly generated different task sets from the MiBench and PARSEC benchmarks with random periods. Fig. 7a and Fig. 7b depict the power profile of compared methods in realistic and worst-case scenarios, respectively. It can be observed that our proposed method and P2AEM keep the peak power consumption beneath the TDP constraint. However, the RPPA method does not consider the peak power constraint and consumes more power than the TDP constraint. Evaluation results in peak power consumption provided in Fig. 8a and Fig. 8b demonstrate that the P2AEM and our proposed method, with different utilizations/workloads and in both realistic and worst-case scenarios, keep the peak power consumption consistently below the TDP constraint. However, the RPPA method has on average 14.65% (up to 37.90%) more peak power consumption. Also, Fig. 9 presents the energy consumption evaluation in both realistic and worst-case scenarios. As shown in Fig. 9, our proposed method, relative to existing methods, can perform better energy efficiency. Our proposed method reduces energy consumption compared to the P2AEM by an average of 10.36% and up to 36.03%. Our simulation results show that the superiority of our method over other methods can be seen in the realistic scenario, and this is what we expected. Because as we said previously, our proposed method delays the execution of the backup tasks until the fault is detected in the system, which reduces the energy consumption on average by 5.92% and up to 14.03% compared to the RPPA method.

6.4 Schedulability Analysis

The overlap elimination in the execution of the primary and backup jobs may lead to an increase in the execution time of the tasks. Therefore, the real-time constraint might be violated, and task mapping/allocation and scheduling policy cannot find feasible scheduling for the tasks. Also, the number of backup tasks executed can affect energy consumption negatively. In this regard, we evaluate the schedulability of the methods, which is defined as feasible task mapping/allocation and scheduling with a limited energy budget. We have considered five different fault rates, and Fig. 10 demonstrates the schedulability evaluation results. It can be observed that our proposed method provides different schedulability while considering the goal of reducing energy consumption at different fault rates. When the fault rate is low, our proposed method reduces energy consumption and increases schedulability by eliminating the execution of backup tasks that are not required to be executed. On the other hand, when the fault rate increases, the decrease of schedulability occurs with an eye to reduce energy consumption.

6.5 Fault Tolerance Analysis

Our proposed method, along with efficiency in power/energy management and ensuring real-time constraint, provides a higher acceptable reliability level. Fig. 11 depicts the reliability and availability analysis of the proposed method compared to the state-of-the-art methods. We have

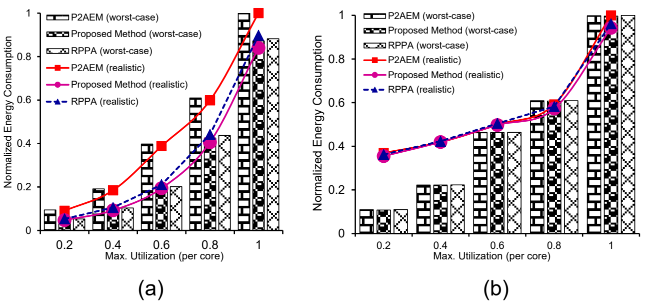

Fig 9. Energy analysis in two scenarios (MiBench (scenario 1) and PARSEC (scenario 2)), a) scenario 1, and b) scenario 2.

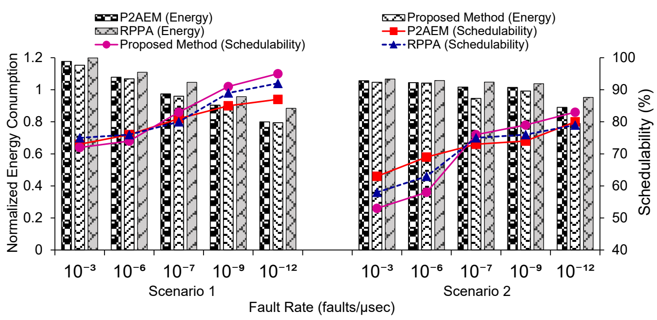

Fig 10. Schedulability analysis.

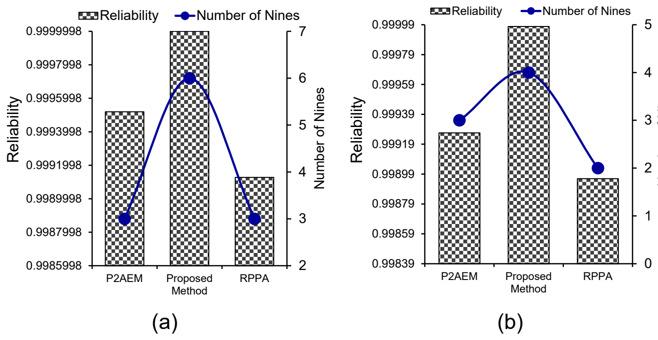

Fig 11. Reliability and availability analysis, a) DVFS is disabled, and b) DVFS is enabled.

considered two scenarios for the evaluation of reliability. First, we considered the DVFS-disabled scenario, and Fig. 11a depicts the simulation results. Then we evaluated the reliability level in the DVFS-enabled scenario, and Fig. 11b illustrates our comparison. We have assumed that the P2AEM method executed the backup tasks in the HP island. It is observed that considering the passive primary/backup technique by providing a suitable task mapping/allocation method, similar to our proposed method, improves the reliability and availability of the system in comparison to core pair and alternative selection mechanisms. Also, it can be seen that the DVFS technique has a negative effect on the reliability of the system. However, our proposed method can provide a higher reliability level than state-of-the-art methods.

7 Conclusion and Future Work

In this paper, we have proposed a mixed passive primary/backup technique that maps and schedules the primary and backup jobs on all cores by considering the fault incidence. Also, the proposed method preserves the peak power consumption below the TDP constraint while meeting the real-time requirement and guaranteeing a high level of reliability. Evaluations in different scenarios show that our proposed method, while ensuring real-time and reliability requirements, can also be effective in reducing peak power and energy consumption and improving the schedulability in realistic execution scenarios.

Today, Machine learning (ML) has gradually become the core component of wide applications in embedded systems. ML-based methods can potentially adapt to modifications in system conditions and workloads, learn from conditions that have occurred in previous periods, and analyze what has already come about, thereby improving their control decisions in reaction to environmental changes. To this end, we will focus on extending our work with the aid of Machine Learning techniques in the future.

References

[1] L. Niu and D. B. Rawat, “Energy-Constrained Standby-Sparing for Weakly Hard Real-Time Systems,” in *2020 IEEE Real-Time Systems Symposium (RTSS)*, Houston, TX, USA, pp. 257-269, 2020.

[2] C. M. Krishna, “Fault-tolerant scheduling in homogeneous real-time systems,” in *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–34, Apr. 2014.

[3] J. Perez Rodriguez and P. Meumeu Yomsi, “Thermal-Aware Schedulability Analysis for Fixed-Priority Non-preemptive Real-Time Systems,” in *IEEE Real-Time Systems Symposium (RTSS)*, pp. 154-166, 2019.

[4] L. Niu and D. Zhu, “Fixed-Priority Scheduling for Reliable and Energy-Aware (m,k)-Deadlines Enforcement with Standby-Sparing,” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

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

[6] A. Roy, H. Aydin, and D. Zhu, “Energy-aware standby-sparing on heterogeneous multicore systems,” in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, USA, pp. 1-6, 2017.

[7] A. Yeganeh-Khaksar, M. Ansari, S. Safari, S. Yari-Karin, and A. Ejlali, “Ring-DVFS: Reliability-Aware Reinforcement Learning-Based DVFS for Real-Time Embedded Systems,” in *IEEE Embedded Systems Letters*, 2020.

[8] 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*, Vienna, Austria, pp. 349-356, 2011.

[9] B. Pourmohseni, F. Smirnov, H. Khdr, S. Wildermann, J. Teich and J. Henkel, “Thermally Composable Hybrid Application Mapping for Real-Time Applications in Heterogeneous Many-Core Systems,” in *IEEE Real-Time Systems Symposium (RTSS)*, pp. 220-232, 2019.

[10] E. Dubrova, “Fault-Tolerant Design.,” in *Springer New York*, 2013.

[11] J. Kim, G. Bhatia, R. Rajkumar and M. Jochim, “SAFER: System-level Architecture for Failure Evasion in Real-time Applications,” in *IEEE 33rd Real-Time Systems Symposium (RTSS)*, pp. 227-236, 2012.

[12] L. Han, L. Canon, J. Liu, Y. Robert, and F. Vivien, “Improved Energy-Aware Strategies for Periodic Real-Time Tasks under Reliability Constraints,” in *IEEE Real-Time Systems Symposium (RTSS)*, pp. 17-29, 2019.

[13] A. Roy, H. Aydin, and D. Zhu, “Energy-aware primary/backup scheduling of periodic real-time tasks on heterogeneous multicore systems,” in *Sustainable Computing: Informatics and Systems*, vol. 29, p. 100474, Mar. 2021.

[14] J. Huang, R. Li, X. Jiao, Y. Jiang, and W. Chang, “Dynamic DAG Scheduling on Multiprocessor Systems: Reliability, Energy, and Makespan,” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3336-3347, 2020.

[15] B. Hu, Z. Cao, and M. Zhou, “Energy-Minimized Scheduling of Real-Time Parallel Workflows on Heterogeneous Distributed Computing Systems,” in *IEEE Trans. on Services Computing*, 2021.

[16] G. Xie et al., “Minimizing Redundancy to Satisfy Reliability Requirement for a Parallel Application on Heterogeneous Service-Oriented Systems,” in *IEEE Transactions on Services Computing*, vol. 13, no. 5, pp. 871-886, 1 Sept.-Oct. 2020.

[17] H. Chniter, O. Mosbahi, M. Khalgui, M. Zhou, and Z. Li, “Improved Multi-Core Real-Time Task Scheduling of Reconfigurable Systems With Energy Constraints,” in *IEEE Access*, vol. 8, pp. 95698-95713, 2020.

[18] X. Wang et al., “Dynamic Low-Power Reconfiguration of Real-Time Systems With Periodic and Probabilistic Tasks,” in *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 258-271, Jan. 2015.

[19] H. Yuan, M. Zhou, Q. Liu, and A. Abusorrah, “Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds,” in *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 5, pp. 1380-1393, September 2020.

[20] Q. -H. Zhu, H. Tang, J. -J. Huang, and Y. Hou, “Task Scheduling for Multi-Cloud Computing Subject to Security and Reliability

Constraints," in *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, pp. 848-865, April 2021.

[21] M. Kang, C. Wen, and C. Wu, "A model predictive scheduling algorithm in real-time control systems," in *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 2, pp. 471-478, Mar. 2018.

[22] A. Ejlali, B. M. Al-Hashimi, and P. Eles, "A standby-sparing technique with low energy-overhead for fault-tolerant hard real-time systems," in *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis (CODES+ISSS '09)*, ACM, pp. 193-202, October 2009.

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

[24] A. Ejlali, B. M. Al-Hashimi, and P. Eles, "Low-Energy Standby-Sparing for Hard Real-Time Systems," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 3, pp. 329-342, 2012.

[25] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware Standby-Sparing Technique for periodic real-time applications," in *IEEE 29th International Conf. on Computer Design*, pp. 190-197, 2011.

[26] P. Purushothaman Nair, A. Sarkar, and S. Biswas, "Fault-tolerant Real-time Fair Scheduling on Multiprocessor Systems with Cold-standby," in *IEEE Transactions on Dependable and Secure Computing*, 2019.

[27] Y. Guo, D. Zhu, H. Aydin, J.-J. Han, and L. T. Yang, "Exploiting primary/backup mechanism for energy efficiency in dependable real-time systems," in *Journal of Systems Architecture*, vol. 78, pp. 68–80, 2017.

[28] Y. Zhang, "Energy-aware mixed partitioning scheduling in standby-sparing systems," in *Computer Standards & Interfaces*, vol. 61, pp. 129–136, Jan. 2019.

[29] A. Roy, H. Aydin, and D. Zhu, "Energy-efficient primary/backup scheduling techniques for heterogeneous multicore systems," in *Eighth International Green and Sustainable Computing Conference (IGSC)*, Orlando, FL, pp. 1-8, 2017.

[30] 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," in *IEEE Access*, vol. 8, pp. 142843-142857, 2020.

[31] P. P. Nair, R. Devaraj and A. Sarkar, "FEST: Fault-Tolerant Energy-Aware Scheduling on Two-Core Heterogeneous Platform," in *2018 8th International Symposium on Embedded Computing and System Design (ISED)*, pp. 63-68, 2018.

[32] M. Ansari et al., "Thermal-Aware Standby-Sparing Technique on Heterogeneous Real-Time Embedded Systems," in *IEEE Transactions on Emerging Topics in Computing*, 2021.

[33] H. Sobhani, S. Safari, J. Saber-Latibari, and S. Hessabi, "REALISM: Reliability-aware energy management in multi-level mixed-criticality systems with service level degradation," in *Journal of Systems Architecture*, vol. 117, p. 102090, Aug. 2021.

[34] 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 Multicore Systems," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 4, pp. 646-654, April 2021.

[35] H. Zamani, D. Tripathy, L. Bhuyan, and Z. Chen, "SAOU: safe adaptive overclocking and undervolting for energy-efficient GPU computing," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '20)*, New York, NY, USA, pp. 205–210, August 2020.

[36] L. A. Johnson, "DO-178B: Software considerations in airborne systems and equipment certification," in *Radio Technical Commission for Aeronautics (RTCA)*, 1992.

[37] 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.

[38] A. Meixner, M. E. Bauer, and D. Sorin, "Argus: Low-Cost, Comprehensive Error Detection in Simple Cores," in *40th Annual IEEE/ACM Int. Symp. on Microarchitecture (MICRO)*, Chicago, IL, pp. 210-222, 2007.

[39] J. Bank and F. Werner, "Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties," in *Mathematical and Computer Modelling*, vol. 33, no. 4–5, pp. 363–383, Feb. 2001.

[40] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," in *The Computer Journal*, vol. 29, no. 5. Oxford University Press (OUP), pp. 390–395, May 01, 1986.

[41] N. Binkert, et.al, "The gem5 simulator," in *ACM SIGARCH Computer Architecture News*, vol 39, no. 2, pp. 1–7, 2011.

[42] S. Li, et.al, "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, 2009.

[43] 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," in *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, pp. 3-14, 2011.

[44] C. Bienia, S. Kumar, J. P. Singh and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 72-81, 2008.

Sina Yari-Karin received his M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2021, and his B.Sc. degree from the Ferdowsi University of Mashhad in 2017. His research interests are embedded system design, low power system design, fault-tolerant system design, cyber-physical systems, real-time systems, and computer architecture.

Roozbeh Siyadatzadeh received the B.Sc. degree from Persian Gulf University, Bushehr, Iran, in 2020. He is Admitted as an Exceptional Talented Student at Sharif University of Technology for M.Sc. programs in 2020. He is currently a member of Embedded Systems Research Laboratory (ESR-LAB) at the department of computer engineering, Sharif University of Technology. His research interests include embedded systems and cyber-physical systems, low-power design, and machine learning.

Mohsen Ansari received his Ph.D. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2021. He is currently an assistant professor of computer engineering at the Sharif University of Technology, Tehran, Iran. He was a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany, from 2019 to 2021. His research interests include cyber-physical systems and hybrid systems design.

Alireza Ejlali received the Ph.D. degree in computer engineering from the Sharif University of Technology in Tehran, Iran, in 2006. He is currently an associate professor of computer engineering at the 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. His research interests include low power design, fault tolerance, real-time embedded systems, and Internet of Things.