# Thermal-Aware Standby-Sparing Technique on Heterogeneous Real-Time Embedded Systems

Mohsen Ansari, Sepideh Safari, Sina Yari-Karin, Pourya Gohari-Nazari, Heba Khdr, Muhammad Shafique, *Senior Member*, *IEEE*, and Jörg Henkel, *Fellow*, *IEEE*, and Alireza Ejlali

**Abstract**—Low power consumption, real-time computing, and high reliability are three key requirements/design objectives of real-time embedded systems. The standby-sparing technique can improve system reliability while it might increase the temperature of the system beyond safe limits. In this paper, we propose a thermal-aware standby-sparing (TASS) technique that aims at maximizing the Quality of Service (QoS) of soft real-time tasks, which is defined as a function of the finishing time of running tasks. The proposed technique tolerates permanent and transient faults for multicore real-time embedded systems while meeting the Thermal Safe Power (TSP) as the core-level power constraint, which avoids thermal emergencies in on-chip systems. Executing the main and backup tasks on the cores at any power consumption below TSP guarantees that no thermal violation occurs. Our TASS proposed method provides an opportunity to remove the overlaps of the execution of main and backup tasks to prevent extra power consumption due to applying the fault-tolerant technique. Meanwhile, in order to maximize the QoS, we employ a heterogeneous platform to execute the main tasks as soon as possible on high-performance cores with more power budget. The backup tasks are executed on low power cores after finishing the main tasks. In this case, when the main task finishes successfully, the whole of its corresponding backup task can be dropped, resulting in a significant amount of power and temperature reduction. Therefore, in the fault-free scenarios, the spare cores can be powered down, and only the main tasks are scheduled and executed on the primary cores. Experiments show that our proposed method improves QoS up to 39.78% (on average by 18.40%) and reduces the peak power consumption and temperature by up to 40.21% and 15.47˚C (on average 28.31% and 13.60˚C), respectively, at runtime, while keeping the system reliability at the required level.

**Index Terms**— Thermal Management, Power Consumption, Standby-Sparing, Real-Time Embedded Systems, QoS, Thermal Safe Power.

## 1 Introduction

CHIP manufacturers have introduced Thermal Design Power (TDP) as the chip-level power constraint for a specific chip used in multicore real-time embedded systems [1][2]. Several hardware-level and software-level techniques are proposed to dissipate the power consumption up to TDP, e.g., cooling techniques as the hardware-level technique and peak-power-aware scheduling policies as the software-level techniques [2][3]. TDP, as the power constraint of the system, could be either pessimistic or thermally unsafe [4]. TDP as the pessimistic constraint is not the maximum accessible power that can be consumed on the chip and can result in significant performance losses [4][5]. On the other hand, if TDP as a chip-level power constraint is not a pessimistic constraint, TDP can be thermally unsafe and can lead to thermal violations [4][5]. In order to avoid thermal violations, Dynamic Thermal Management (DTM) technique is employed on the chip to throttle down the cores using Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) [2][5][37]. However, the DVFS technique might lead to missing the deadlines of the real-time tasks and degrading the system reliability due to increasing the worst-case execution time, and hence, this is not acceptable in real-time embedded systems [7][8]. Therefore, in this paper, we employ the Thermal Safe Power (TSP) [4] as the per-core power constraint, which is defined as a function of the number of simultaneously operating cores [4]. Executing cores at any power consumption below TSP guarantees to avoid thermal violations, and thereby DTM will not be triggered [36].

In addition to timeliness requirements of the tasks, Quality of Service (QoS) requirements must be considered in multicore real-time embedded systems. The QoS requirement refers to the utility function of tasks after they proceeded. It should be noted that the QoS-aware real-time embedded systems should incorporate inherent high reliability features to tolerate different types of faults [2]. This

- *M. Ansari, S. Safari, S. Yari-Karin, P. Gohari-Nazari, and A. Ejlali are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (e-mail: {mansari, ssafari sinayari, gohary}@ce.sharif.edu, and ejlali@sharif.edu).*
- *M. Ansari, S. Safari, H. Khdr, and J. Henkel are with the Karlsruhe Institute of Technology, Karlsruhe 76131, Germany (e-mails: {Mohsen.ansari, Sepideh.safari; Heba.Khdr, Henkel}@kit.edu).*
- *M. Shafique is with the Division of Engineering, New York University Abu Dhabi (NYU AD), Abu Dhabi, United Arab Emirates (e-mail: muhammad.shafique@nyu.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.)*
*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*

is because the system must guarantee that each task is executed within its deadline, even in the presence of different faults [10]. For example, in the signal processing systems, although missing the deadline for some tasks may not result in failure, the outdated or half-baked processed data may be useless for users [13]. Therefore, the system must guarantee its functionality and maximize the QoS of the tasks even when faults occur. Consequently, employing a system-level fault-tolerant technique for such systems is mandatory. Some fault-tolerant techniques like Checkpointing and task re-execution can just tolerate transient faults, but the standby-sparing technique is a common fault-tolerant approach to deal with both transient and permanent faults. Two well-known standby-sparing techniques are Hot Standby-Sparing (HSS) and Cold Standby-Sparing (CSS). In the HSS technique, both primary and spare cores are active simultaneously, but in the CSS technique, the spare core is inactive and is activated when a fault occurs on the primary core. For low power system design, it is preferable to avoid as far as possible the simultaneous execution of the main and backup tasks for preventing peak power escalation, and hence, the CSS technique can be more suitable. However, since in real-time embedded systems, meeting deadlines is one of the requirements, using the CSS technique in a straightforward way may not be a feasible solution. Therefore, the CSS technique should be exploited intelligently because otherwise, it imposes extra time overhead. Moreover, although the CSS technique has less power overhead compared to HSS and N-Modular Redundancy (NMR) [2][8], it still might lead to violating the power and thermal constraints.

This paper proposes a Thermal-Aware Standby-Sparing technique (called TASS) that exploits the CSS technique on the heterogeneous multicore platforms to maximize the quality of service for soft real-time tasks while at the same time meeting the Thermal Safe Power (TSP) constraint. Our proposed method tries to schedule the backup tasks after finishing the main tasks on the spare cores to remove the overlaps between the main and backup tasks. In this case, when the main task finishes successfully, the whole of its corresponding backup task can be dropped, resulting in a significant amount of power reduction.

**The main contributions of this work are:**

- A thermal-aware standby-sparing system that maximizes QoS of soft real-time tasks for real-time embedded systems.
- A mapping scheme that assigns soft real-time tasks to core pairs of the heterogeneous multicore platform (i.e., each core pair has one primary core and one spare core) and balances the utilization of core pairs and determines the voltage-frequency levels of the cores, keeping the power consumption of each underlying core under the TSP.
- Employing Thermal Safe Power (TSP) as the power constraint instead of TDP. This is because using TDP as the power constraint of a system might lead to thermal violations and thereby increasing the probability of missing deadlines.

In order to compare our TASS method with state-of-the-art methods, we have performed full system simulations using well-established frameworks like gem5 [15], McPAT [16], HotSpot [17], and TSP [4]. Our experiments show that TASS provides up to 40.21% (on average by 28.31%) peak power reduction compared to state-of-the-art methods. Also, our TASS method achieves up to 15.47˚C temperature reduction compared to other methods while keeping the system reliability at a required level.

The following motivational example shows how our proposed TASS method can satisfy timing and core-level power constraints compared to one of the state-of-the-art methods.

### 1.1 Motivational Example

This example provides some insight on the disadvantage of using the constant chip-level power constraint in fault-tolerant systems. For simplicity of presentation, consider a heterogeneous multicore system with two islands which each of them has two homogenous cores. In this example, the system executes a synthetic application task graph with nine dependent tasks $\{T_1, \ldots, T_9\}$. Fig. 1a shows dependencies between the tasks where each task has a triple of parameters $\{WC_{LO}, WC_{HI}, D_i\}$, in which $WC_{LO}$ and $WC_{HI}$ are the worst-case execution time on the low-power and high-performance cores, respectively, and $D_i$ is the dedicated deadline of $T_i$. The peak power consumption of the tasks on different types of cores is shown in Table 1. In this example, we compare our proposed method with a TMR system (i.e., NMR system with $N$=3) where each task has three copies, and the result of them is compared to perform a complete majority voting.

Fig. 1 shows two possible schedules where meet different power constraints, while one of them misses the timing and temperature constraints. In Fig. 1b and Fig. 1c, a peak-power-aware scheduling method is shown that meets different values of TDP. This method exploits N-Modular Redundancy (NMR) to improve system reliability. In this method, from the beginning of the execution, at first, two copies of each task are mapped and scheduled on cores based on the lowest utilization policy such that the chip TDP is met [2]. For scheduling the last copies, starting from

Table 1. The peak power consumption of tasks running on different types of cores

| High Performance Island | | Lowe Power Island | |
|---|---|---|---|
| Task Name | Power value | Task Name | Power value |
| $T_1$ | 2W | $T_1$ | 1W |
| $T_2$ | 3W | $T_2$ | 2W |
| $T_3$ | 2W | $T_3$ | 1W |
| $T_4$ | 2W | $T_4$ | 1W |
| $T_5$ | 4W | $T_5$ | 3W |
| $T_6$ | 2W | $T_6$ | 1W |
| $T_7$ | 3W | $T_7$ | 2W |
| $T_8$ | 2W | $T_8$ | 1W |
| $T_9$ | 2W | $T_9$ | 1W |

Table 2. The core-level power constraint information of the cores at different situations when the number of active cores are different on different islands

| **High Performance Island** | | **Lowe Power Island** | |
|---|---|---|---|
| Number of active cores | The core-level Power constraint | Number of active cores | The core-level Power constraint |
| 1 | 5W | 1 | 3W |
| 2 | 3W | 2 | 1W |

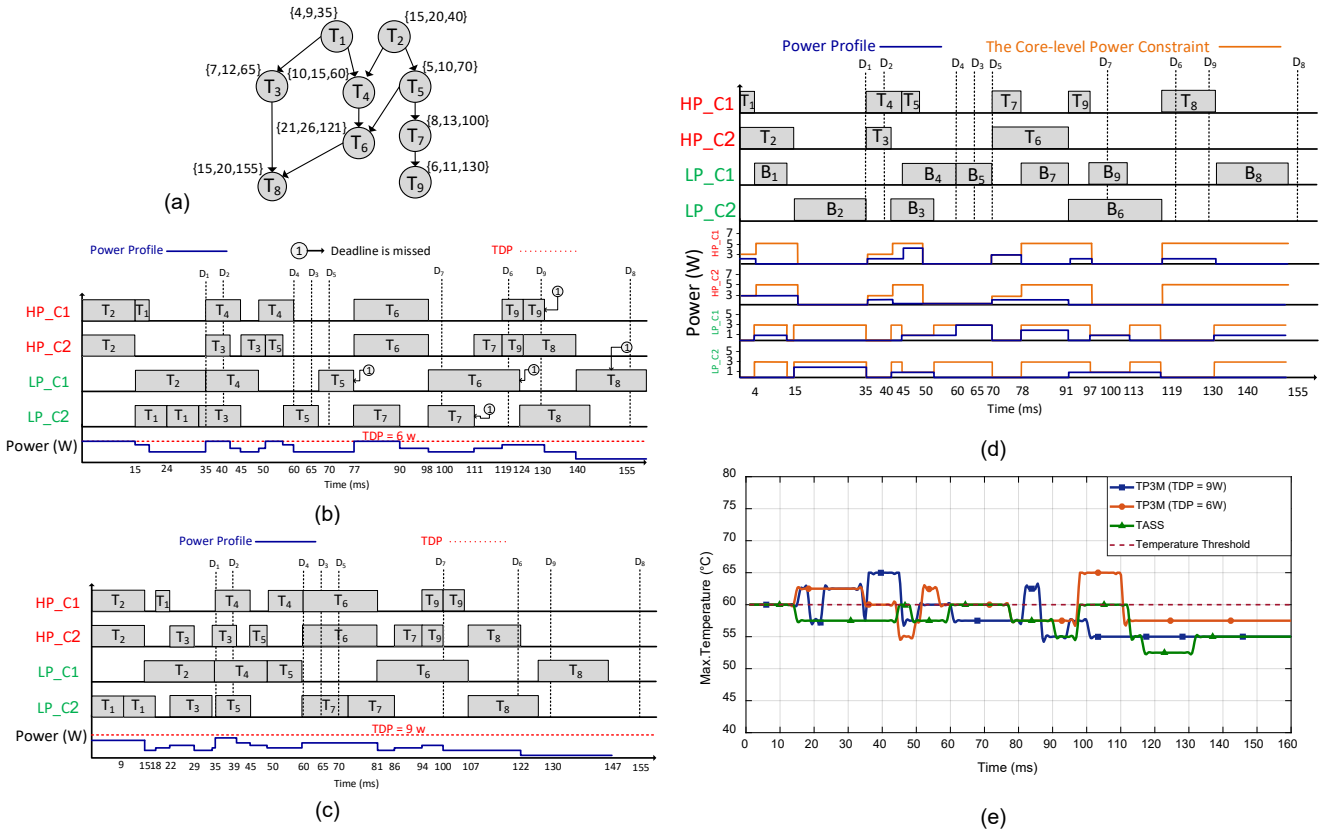


Fig. 1. The motivational example for showing advantages of using the core-level power constraint instead of TDP and standby-sparing instead of TMR on a heterogeneous multicore system with two islands which each of them has two homogenous cores; (a) an application tasks graph with nine tasks; Scheduling the tasks according to the TP3M method [2] in the worst-case fault-scenario (b) by considering a lower-case TDP constraint; (c) by considering a upper-case TDP constraint; (d) Applying our proposed method (TASS) in the worst-case fault scenario; (e) Maximum temperature profile of different methods.

the end of the execution of two other copies of the same task, the third copy is scheduled such that the chip TDP is met [2]. In Fig. 1b and Fig. 1c, we assume that the chip TDP is equal to 6W and 9W, respectively. As seen in these figures, when the chip TDP of 6W is considered (see Fig. 1b), as a pessimistic constraint, which is not the maximum accessible power that can be consumed on the chip, it can result in both deadline and thermal violations (see Fig. 1e). On the other hand, when the chip TDP of 9W is considered (see Fig. 1c) as an upper-case constraint (not a pessimistic constraint), it does not lead to deadline violations; however, thermal violations still happen (see Fig. 1e).

In Fig. 1d, we consider a cold standby-sparing system where only two copies of each task are executed on the system such that the core-level power constraint is met. To do this, we consider two core pairs so that each core pair is included one low-power core and one high-performance core. In our proposed method, we employ a core-level power constraint, which we empirically set its value to avoid thermal violations. The core-level power constraint information of the cores is shown in Table 2. This table shows the value of the power constraint that the cores can consume when the number of active cores is different. Note that for simplicity of presentation, we have calculated the core-level power constraint values for the number of active cores per island individually. However, in the rest of this paper, when we present our method, and in the evaluation section, we consider that the numbers of active cores in both islands are required together to calculate the core-level power constraint for any type of cores; i.e., we employ Thermal Safe Power (TSP). Fig. 1d shows our proposed method such that it meets the core-level power and timing constraints simultaneously. The proposed method executes the backup tasks after finishing the main tasks on the spare core to delete the overlapping execution between the main and the backup tasks. For example, at first, we schedule $T_1$ on HP_C1, and then after finishing the mentioned main task, the backup task of $T_1$ is scheduled on LP_C1. This leads to turning off all of the spare cores at fault-free execution scenario.

It is worthy to mention that the backup task can be managed in different ways like Hot Standby-Sparing (HSS), Cold Standby-Sparing (CSS), and Warm Standby-Sparing (WSS). In HSS and WSS, we should execute the backup tasks in the overlap form with the execution of the main tasks, and therefore, the peak power consumption and the temperature increase in all fault scenarios.

**Paper Organization**: The rest of this paper is formed as follows. Section 2 reviews related work and Section 3 presents models and assumptions. In Section 4, we present the details of our proposed method. The experimental results of the proposed method are shown in Section 5, and we conclude the paper in Section 6.

## 2 Related Work

Researches related to TASS are discussed here. Ejlali et al. in [23] have proposed a hardware redundancy technique for dual-core real-time systems that execute frame-based tasks without any dependencies. In this work, the primary core uses DVS, and the spare core uses DPM. It should be noted that the DVS method is used for energy management, and it has a negative impact on reliability. Also, the mentioned method uses an analytical approach to assign the supply voltage value of the primary core at runtime to reduce energy consumption by exploiting dynamic slack times and meet the reliability target. In [24], the standby-sparing technique is used for reliability improvement for fixed-priority periodic real-time tasks. Indeed in [24], the SSFP algorithm has proposed to delay the execution of the backup task as much as possible by using a dual-queue mechanism to reduce energy consumption. Roy et al. in [25] have used a standby-sparing technique for heterogeneous multicore systems included high-performance and low-power cores. The proposed method in [25] has considered the challenge of heterogeneous multicore system design and has answered two questions. First, which type of core should be selected for main and backup tasks to minimize energy consumption? Secondly, how much should the frequency of the primary core be for the purpose of energy reduction? Also, the authors have shown that the selection of LP (Low Power) or HP (High Performance) cores and allocation of frequency on the primary core has a significant impact on energy consumption. The proposed standby-sparing method in [8] employs EDF scheduling on the primary core and EDL scheduling on the spare core for periodic real-time tasks to execute main tasks as soon as possible and delay the execution of backup tasks. The standby-sparing method proposed in [27] exploits task partitioning and assigns the frequency of tasks at runtime to hold energy consumption at a minimum level and meet fault-tolerant constraints for real-time tasks on heterogeneous multicore systems. Moreover, the proposed method in this paper takes into account the different execution time and power parameter scaling factor.

The works [28] and [29] have proposed the Paired-Standby-Sparing (Paired-SS) and Generalized-Standby-Sparing (Generalized-SS) methods based on the standby-sparing technique. 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 two categories, primary cores and secondary cores. The main 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. Also, Preference-Oriented Earliest Deadline (POED) scheduler has been studied, and experimental results have shown that POED-based methods perform better than SS-based methods in terms of energy, especially for high-loaded systems. The proposed three-layer framework in [30] 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, real-time constraints are considered in the feedback system. The proposed method in [31] is a low-energy standby-sparing system. The proposed method 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. In [32], the authors have proposed a peak-power-aware standby-sparing technique for periodic real-time tasks to keep peak power consumption under TDP constraint. In this method, the primary and spare cores exploit PPA-EDF and PPA-EDL, respectively. The proposed method delays the execution of backup tasks as much as possible and tries to cancel the execution of backup tasks to reduce power consumption. As discussed in the previous sentences, none of the mentioned works consider a temperature threshold.

Table 3. Summary of state-of-the-art works focusing on different types of standby-sparing with different objectives

| # of Reference | Task Model | Architecture | | | | Technique(s) | | |
|---|---|---|---|---|---|---|---|---|
| | | Dual-Core | Multicore | Homogeneous | Heterogeneous | Objective | Power Constraint | SS type |
| [23] | Frame-Based | ✓ | | ✓ | | Energy | - | HSS |
| [24] | Periodic | ✓ | | ✓ | | Energy | - | CSS |
| [25] | Frame-Based | ✓ | | | ✓ | Energy | - | HSS |
| [8] | Periodic | ✓ | | ✓ | | Energy | - | HSS |
| [27] | Frame-Based | | ✓ | | ✓ | Energy | - | HSS |
| [28] | Periodic | | ✓ | ✓ | | Energy | - | HSS |
| [29] | Task Graph | ✓ | | ✓ | | Energy | - | HSS |
| [30] | Periodic | | ✓ | ✓ | | Energy | - | HSS |
| [31] | Frame-Based | ✓ | | ✓ | | Energy | - | HSS |
| [32] | Periodic | | ✓ | ✓ | | Power | TDP | HSS |
| TASS | Task Graph | | ✓ | | ✓ | QoS and Power | TSP | CSS |

Table 3 summarizes the state-of-the-art works discussed in this paper. The works in [8][23][25]-[31] focus on Hot Standby-Sparing (HSS) systems for reducing energy consumption. The work in [24] focuses on Cold Standby-Sparing (CSS) systems for reducing energy consumption. Finally, only [32] has considered meeting TDP in HSS systems for periodic real-time tasks. Also, in Table 3, we show the features of TASS compared to the state-of-the-art works.

In summary, the previous works in the context of standby-sparing embedded systems did not consider thermal violations on the chip. In this paper, we propose a cold standby-sparing system where the main tasks and the backup tasks are scheduled on the primary and spare cores, respectively, such that the Thermal Safe Power (TSP) and real-time constraints are met.

## 3 Models and Preliminaries

In this section, we introduce the system, task, power consumption, and fault models used in this paper.

### 3.1 System and Power Consumption Model

We consider a heterogeneous multicore system that consists of High_Performance (HP) cores and Low_Power (LP) cores. In this paper, we consider that the system consists of $k=m/2$ core pairs CP={$CP_1$, $CP_2$, …, $CP_k$}. Each core pair consists of one primary core (HP core) and one spare core (LP core). When a task ($T_i$) is mapped to the primary core, a backup copy ($B_i$) is also mapped to the corresponding spare core.

The power consumption of our TASS system consists of static and dynamic power components in which the static power ($P_{static}$) is dominated by the leakage current, and the dynamic power ($P_{dynamic}$) is commonly dissipated due to system activity [7][9] (See Eq. 1).

$$P_{total}(V_i, f_i) = P_{static} + P_{dynamic} = I_0 e^{\frac{-V_{th}}{\eta V_T}} V_i + \alpha_i C_L V_i^2 f_i \quad (1)$$

Where $C_L$ is the switched capacitance, $\eta$ is a technology parameter, $V_i$ and $f_i$ are supply voltage and operational frequency, and ($\alpha_i$) is the activity factor for the task $T_i$. Due to the heterogeneity nature of our proposed system, each task exhibits different power consumption characteristics on different types of cores. Therefore, we use the superscripts HP and LP cores to distinguish between the values of the task power consumption parameters on the HP and LP cores, respectively (e.g., $\alpha_i^{HP}$ and $\alpha_i^{LP}$).

Under the DVFS technique, the voltage $V_i$ used for the execution of the tasks on a core may be less than the maximum voltage $V_{max}$. We denote the normalized voltage $\rho_i$ as [10]:

$$\rho_i = \frac{V_i}{V_{\max}} \quad (2)$$

where $V_{max}$ is the maximum voltage corresponding to the maximum frequency $f_{max}$. Note that according to the almost linear relationship between voltage and frequency [7][8][10], we can write: $\rho_i = V_i/V_{max} = f_i/f_{max}$. Therefore, Eq. 1 can be rewritten as:

$$P_{total}(V_i, f_i) = \rho_i (I_0 e^{\frac{-V_{th}}{\eta V_T}} V_{\max}) + \rho_i^3 (\alpha_i^{HP\_or\_LP} C_L V_{\max}^2 f_{\max}) \quad (3)$$

### 3.2 Task Model

In this paper, task graphs with soft real-time requirements are considered, where each task graph is a directed acyclic graph (DAG) and has $n$ dependent non-preemptive tasks $\Phi$={$T_1$, $T_2$, …, $T_n$} with the same criticality level [2][20][39][40][41][42]. In a task graph, a task is scheduled only after that all its predecessors are executed completely. The result of a predecessor task is passed to its successors upon completion. Each task $T_i$ in the mentioned task graph has a triple of parameters {$wc_i^{HP}$, $wc_i^{LP}$, $D_i$}:

- $wc_i^{LP}$: The worst-case execution time on the low-power cores at the maximum supply voltage and operational frequency.
- $wc_i^{HP}$: The worst-case execution time on the high-performance cores at the maximum supply voltage and frequency.
- $D_i$: the dedicated deadline of $T_i$.

It should be noted that since the main and backup tasks are executed on the HP and LP cores, respectively, $wc_i^{HP}$ and $wc_i^{LP}$ are considered for the main and backup tasks, respectively.

### 3.3 Fault Model

High reliability, as one of the major design objectives in multicore embedded systems, is subjected to different types of faults [6][19]. In this paper, we consider 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 [2][7]. In our evaluations, the transient fault rate is considered as [6][7]:

$$\lambda(V_i) = \lambda_0 \times 10^{\frac{V_{\max} - V_i}{d}} \quad (4)$$

Where $\lambda_0$ is the transient fault rate at $V_{max}$, and $d$ determines the sensitivity of the system to voltage scaling. The functional reliability of a task can be written as [2][7][9][38]:

$$R(T_i) = e^{-\lambda(V_i) \times wc_i} \quad (5)$$

Where $wc_i$ is the worst-case execution time of $T_i$. If the reliability of the main and backup tasks be independent because of the heterogeneity, the reliability of the cold standby-sparing technique for each task is:

$$R(T_i, B_i) = R(T_i) + (1 - R(T_i)) \times R(B_i) \quad (6)$$

The reliability of a standby-sparing system with $n$ tasks executing by our proposed method can be calculated as:

$$R_{system} = \prod_{i=1}^{n} R(T_i, B_i) \quad (7)$$

Note that our standby-sparing system can tolerate a permanent fault for each core pair. For tolerating the transient faults, the standby-sparing system requires a fault detection method. For this purpose, our processing cores exploit a low-cost and low power hardware checker like Argus [12]. In this paper, if during the execution of an original task, the fault has not been detected, its results are applied to the system, and the whole of its corresponding backup task is canceled. However, when a fault occurs, we do not consider the faulty task and continue with its backup task and apply the results of the backup task as the correct results.

## 4 Our Proposed Method

### 4.1 Concept Overview and Our Novel Contribution

This paper proposes a Thermal-Aware Standby-Sparing Technique (called TASS) in heterogeneous multicore real-time embedded systems. It should be noted that using a temperature threshold directly within mapping and scheduling is difficult and complex because when a task is executed on a core, it leads to increase the temperature of that core and the other core temperatures as well. In this paper, we employ Thermal Safe Power (TSP) as the power constraint, which is an abstraction that provides thermally safe power constraint as a function of the number of simultaneously operating cores [4]. Executing cores at any power consumption below TSP guarantees avoiding thermal violations, and thereby DTM will not be triggered. We should

mention that calculating TSP values are independent of different types of running tasks, but when we schedule a task on a time slot we update the TSP values of the cores based on activated the number of cores due to our scheduling. In our method, we map and schedule the tasks such that main tasks are executed as soon as possible on the high performance cores and the backup tasks on the low power cores to reduce the peak power. Our proposed method tries to schedule the backup tasks on the spare cores after the main tasks finish their execution in order to remove the overlapping between the main and backup tasks. In this case, when the main task finishes successfully, the whole of its corresponding backup task can be dropped, resulting in a significant amount of power reduction. The overview of TASS is shown in Fig. 2.

### 4.2 Problem Definition

Given a multicore system with $m$ cores that executes a set of $n$ soft real-time tasks. The problem is how to find the task-to-core assignment, the scheduling of the tasks (the start time $s_i$ and the finish time $f_i$ of all main and backup tasks), and the V-F level of each task to achieve the target goal.

In this formulation, $v$ is the number of available V-f levels for each core. The V-f level assignments and task-to-core mapping are represented by the matrix $X\epsilon\{0,1\}^{n\times m\times v}$. The task $T_i$ is assigned to the core $k$ and is executed under the V-f level $l$ if and only if $X_{ikl} = 1$.

The goal of our method is to maximize the QoS of the system while keeping the power consumption of each core under the TSP constraint ($P_{TSP,core}$). We have formulated the mentioned problem in the following.

**Optimization Goal:** Maximize the QoS of the system defined by the average of the QoS of all tasks. Note that the probability of successful execution of each main task $T_i$ is defined by $P_i$, and the probability of its failure is defined by 1-$P_i$.

$$\text{Maximize } QoS_{sys} = \frac{\sum_{i\in\Phi} P_i\times UF_i(f_{T_i},D_i)+\sum_i(1-P_i)\times UF_i(f_{B_i},D_i)}{n} \quad (8)$$

where $UF_i$ is the utility function of task $T_i$ that describes the utility value of the mentioned task as a function of its finishing time, and $f_i$ is the finish time of $T_i$. In this paper, we have considered the linear function and can be rewritten as [14][34]:

$$UF_i = \begin{cases} 1 & f_i \le D_i \\ \dfrac{(D_i - f_i)}{D_i(x-1)} + 1 & D_i < f_i \le x \times D_i \\ 0 & f_i > x \times D_i \end{cases} \quad (9)$$

where $x$ is a variable bigger than 1 in order to determine the maximum deadline violation and is received by the designer.

**Core Power Constraint:** The power consumption of each underlying core at each time slot $t$ must be less than the core TSP constraint.

$$\forall\, k\colon P_k(t) \le P_{TSP,k}(\#\ active\ cores) \text{ at each time } t \quad (10)$$

**Timing Constraint:** The finish time of each task should be less than its deadline.

$$\forall i\colon f_i \le D_i \quad (11)$$

**Core Assignment Constraint:** Each task can only be mapped to a core.

$$\forall i, l\colon \sum_k X_{ikl} = 1 \quad (12)$$

**Task's Reliability Constraint:** Since we exploit DVFS for tasks, the reliability of tasks should satisfy a minimum reliability threshold after DVFS.

$$e^{-\lambda(V_i)\times\frac{wc_i^{HP\ or\ LP}}{\rho_i}} \ge R_{th} \quad (13)$$

**Dependency Constraint:** Dependency constraints between tasks should not be violated. If there is a dependency between two tasks, the completion time of the previous task should be smaller than the start time of its dependent task.

$$\forall i, j\colon f_i \le s_j, \qquad if\ T_i \text{ is the predecessor of } T_j \quad (14)$$

This problem is known to be an NP-hard problem in a strong sense [7][10][35]. Indeed, finding an optimal solution will have exponential-time complexity. Therefore, we propose a heuristic for mapping and scheduling method the tasks that aim at maximizing the QoS of the system under the aforementioned constraints.

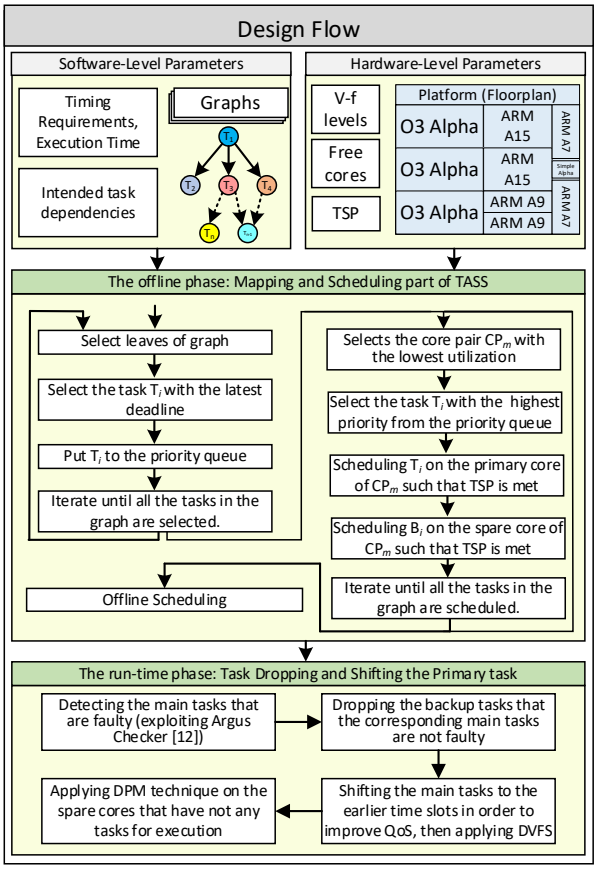


Fig. 2. Overview of our proposed method, TASS.

### 4.3 Algorithm Discussion

The main idea of our proposed scheduling method is to provide an opportunity to remove the overlaps of the execution of main and backup tasks to prevent extra peak power consumption due to applying the cold standby-sparing technique for improving reliability.

The Latest Deadline First (LDF) scheduling policy is an optimal policy that minimizes the maximum lateness of a set of tasks with data dependency and the same arrival times

[14]. It should be noted that EDF is not optimal under dependency constraints because it achieves greater lateness with respect to LDF [14]. LDF makes the scheduling queue from the tail to the head of the graph. This policy selects the task with the latest deadline one by one to set the priority of tasks. This process is repeated until all tasks in the graph are selected and put in the scheduling queue. Then, tasks are selected from the head of the scheduling queue such that the first task inserted in the queue has the last priority and will be executed last, and the last task inserted in the scheduling queue has the first priority and will be executed first.

Algorithm 1 shows the details of our proposed mapping and scheduling method. This algorithm receives a task graph application with all its information, the list of TSPs for all cores at different times, and the set of core pairs, and provides tasks mapping and scheduling in each core pair. Lines 1-7 extract the priority of tasks according to the LDF policy. As it was mentioned before, LDF builds the priority queue from the tail to the head of the graph. Among the tasks that have not successors or whose successors have all been selected, the LDF policy selects the task with the latest deadline to be scheduled last. The algorithm at first selects the leaves of the graph in the temporary queue TQ in line 3. Then, it selects the task with the latest deadline in Line 4. The selected task $T_i$ is put in the priority queue, which is a Double-Ended queue (Line 5). In line 6, the selected task $T_i$ is removed from the graph. This process is repeated until all tasks in the graph are selected and put in the priority queue. It should be noted that the tasks are selected from the head of the priority queue to be scheduled, such that the first task inserted in the priority queue will be scheduled last. Therefore, the algorithm selects the tasks in the priority queue one by one and schedules them.

In lines 8-29, the algorithm schedules the tasks such that the TSP of all cores is met, and the QoS of backup tasks are maximized. To do this, the algorithm iterates until all main and backup tasks are scheduled. In line 9, the algorithm selects the task with the highest priority in the priority queue.

Then, in order to balance the utilization of the core pairs, the algorithm selects the core pair with the lowest utilization (denoted by $\varphi$ in line 10) to map the selected task. In order to schedule the selected task (its main and backup), the algorithm finds the maximum finish time of all predecessors of the selected task in line 11. Then, in lines 12-21, the main task is scheduled such that the power constraint is met. TSP values are computed for all possibilities of the number of active cores using the algorithm proposed in [4]. On the primary core of $\varphi$, the algorithm finds the first free time slot after the finish time of the predecessors of $T_i$. Then, it checks whether the maximum power consumption of $T_i$ is equal or less than the TSP value of the primary core of $\varphi$ at the time slot $t$. If the mentioned condition is satisfied, the algorithm schedules $T_i$ on the time slot $t$ on the schedule of the primary core $\varphi.S_{primary}$. Otherwise, until a satisfactory time slot is found, the algorithm repeats this process. After scheduling the main task, in lines 22-28, the backup task of $T_i$ is scheduled. To do this, we set the variable $k$ to the finish time of the corresponding main task because we want to prevent the overlap between the main and backup tasks. It should be noted that the backup tasks can miss their deadlines because the task model is soft real-time. Therefore, the backup task of $T_i$ is scheduled on the spare core of $\varphi$ such that the power constraint is met.

**Algorithm 1. The mapping and scheduling policy of TASS**

**Inputs:** Φ: The set of tasks in graph, TSP for all possible number of active cores, the set of core pairs CP={$CP_1$, $CP_2$, …, $CP_k$}.
**Output:** The task scheduling $S_i$ on each core $C_j$.

```
BEGIN
1.  TQ = Ø, PQ = Ø;   // Initialize the Temp. and priority queues
2.  while Φ ≠Ø do
3.     TQ = Φ.return();                    // Return leaves of graph
4.     Ti = TQ.select();                   // Return the task with the latest
                                              deadline in PQ
5.     PQ.put(Ti); // Put Ti to the priority queue (a double-
                                                        ended queue)
6.     Φ.remove(Ti);                       // Remove Ti from the graph
7.  end while
8.  while PQ ≠ Ø do
9.     Ti = PQ.select();   // Select Ti from the priority queue PQ
10.    φ = min_utilization{CPm, 1≤m≤k};
11.    k =Max.Finish_time_of_predecessors(Ti);
---    Scheduling Ti on the primary core of CPm
12.    while k ≤ Di-WC_i^HP do
13.       t←Find first free time slot after k on φ.S_primary;
14.       TSP.φ.S_primary = TSP(#active_cores at t); // Return TSP
                                        of φ.S_primary in HP island at t
15.       if Max.Power(Ti) ≤ TSP.φ.S_primary at time t do
16.          Schedule Ti at t on φ.S_primary;
17.          PQ.remove(Ti);
18.          break;
19.       k=t+1;
20.    end while
21.    k = Finish time of Ti on φ.S_primary;
---    Scheduling Bi on the spare core of CPm
22.    while Bi is not scheduled do
23.       t←Find first free time slot after k on φ.S_primary;
24.       TSP.φ.S_spare = TSP(#active_cores at t);  // Return TSP
                                        of φ.S_spare  in LP island at t
25.       if Max.Power(Bi) ≤ TSP.φ.S_spare do
26.          Schedule Bi at t on φ.S_spare;
27.          break;
28.    end while
29. end while
END
```

In runtime, we exploit the opportunities created during the actual execution of the tasks to reduce further power consumption and improve QoS. This can be provided through an online monitor for the system at runtime. The online monitor works to control the execution accuracy of the main tasks to handle the runtime state of the system and apply DPM and DVFS. The online monitor needs to find the main tasks that have finished successfully and cancels the execution of their corresponding backup tasks. Since the fault rate is rare, most of the time, there is no need to be executed the backup tasks. Therefore, as soon as the main task finishes successfully, the online monitor drops the execution of the backup tasks.

For applying the DVFS technique, when the dynamic slacks are released at runtime, our method uses DVFS to

Table 4. The details of system configuration

| Parameter | Configuration | |
|---|---|---|
| | LITTLE island | big island |
| Core Type | ARM Cortex-A7 | ARM Cortex-A15 |
| Machine Type | In-Order | Out-Of-Order |
| Core Volt. And Freq. | [0.78V, 0.4GHz] to [0.93V, 1.4GHz] | [0.89V, 0.8GHz] to [0.9995V, 2GHz] |
| Feature Size | 14 nm | |
| Core Microar-chitecture | ARMv7-A | |
| L1 Cache | 32KB, 8KB block-width, 4-way | |
| L2 Cache | 2MB, 16-way | |
| Memory | 2GB, 32-bit LPDDR3e | |

further reduce the power consumption. Therefore, at runtime, we determine static and dynamic slacks on all HP cores and apply the EVEN-DVFS technique for the main tasks [21]. The EVEN-DVFS technique distributes slacks evenly among all tasks that can use them. The advantage of the EVEN-DVFS technique is its linear time complexity when compared to the quadratic complexity of other DVFS techniques [10]. Like the previous works in the literature, we have considered the time overhead of applying DVFS on the WCET of the tasks [6][7][9][42]. Therefore, when we want to apply DVFS, we add its time overhead to the WCET of tasks. However, this overhead is negligible.

## 5 Experimental Setups And Results

In this section, the effectiveness of TASS is evaluated, employing the well-established frameworks: the gem5 full-system simulator [15], McPAT [16], HotSpot [17], QUILT [33], and TSP [4]. We ran our simulations with various task graphs, including real-life applications of the MiBench benchmark suite [18] running on a heterogeneous multi-core platform, which has two types of cores: (*i*) ARM Cortex-A7 as LP core, (*ii*) ARM Cortex-A15 as HP core. Meanwhile, we considered that the system supports per-core DVFS. The details of simulation configurations for the processing cores of our system are summarized in Table 4. Fig. 3 shows our tool flow for evaluating TASS. We simulate our experiments conducted on gem5 [15] and McPAT [16] for the technology node of 14nm. As shown in Fig. 3, we extract the power consumption, the worst-case execution time, reliability, and temperature values via gem5 and McPAT. The tool QUILT [33] generates the chip floorplan based on the results of McPAT. For extracting the RC-thermal model of the chip, which is necessary to compute TSP

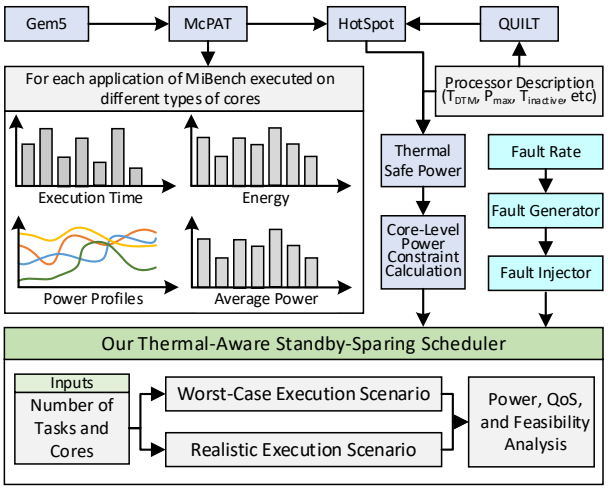


Fig. 3. The tool flow of our proposed method.

[4], we use the HotSpot [17]. Fig. 4 shows the execution time and average and peak power consumption of applications of the MiBench benchmark suite. Then, we employ a system-level simulator for simulating different execution scenarios.

We compare our proposed method with state-of-the-art techniques. The comparison partners in our evaluations are: (*i*) TP3M [2]: This method has proposed a peak-power-aware reliability management approach for N-Modular redundancy, which removes the overlaps of the peak power of concurrently executing tasks to keep the maximum power consumption below the chip TDP; (*ii*) ConvSS [19]: The conventional standby-sparing scheme where each backup task is executed in parallel with its main tasks.

In order to compare TASS with state-of-the-art techniques, we have exploited the Standard Task Graph Set (STG), which is a kind of benchmark for the evaluation of multi-core scheduling algorithms [22]. We considered that the tasks of task graphs were randomly selected from MiBench [18]. In the experiments, each task graph has a different parallelism degree [20]. In our experiments, we consider three classes of task graphs with different parallelism degrees like [2]. For each graph, if $n$ is the number of tasks and $h$ is the height of the task graph, $h$ can vary between 1, which is the highest parallelism degree, and $n$, which is the chained task graph with the lowest parallelism degree. According to the above explanation, we consider the following classes of graphs: 1) task graphs with high parallelism degree $1 \le h < n/3$, 2) task graphs with medium parallelism

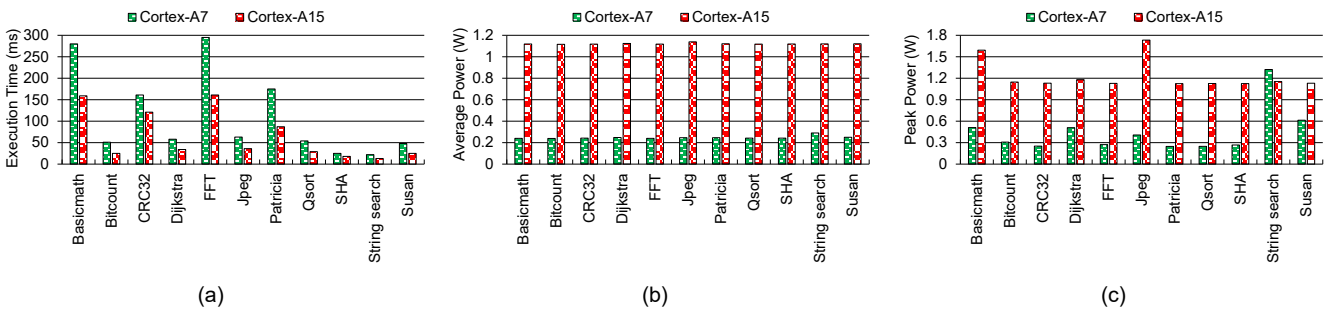


Fig. 4. MiBench benchmark characterization running on ARM Cortex-A7 and ARM Cortex-A15 cores. a) Execution time, b) Average power consumption, and c) Peak power consumption.

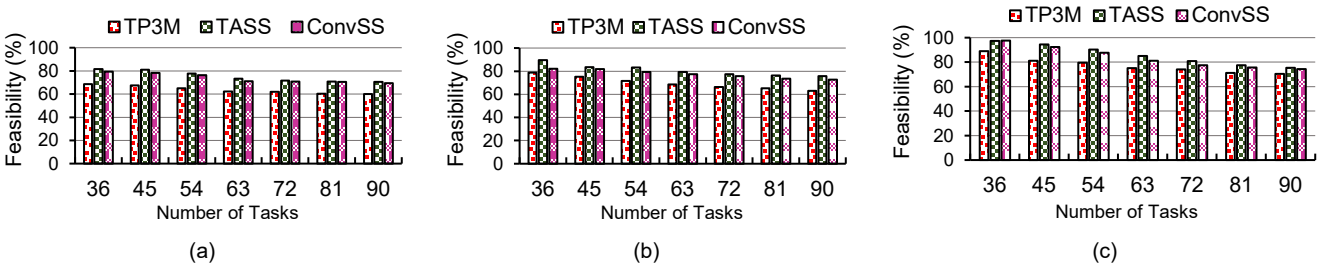


Fig. 5. Feasibility of schemes in the worst-case scenario for different parallelism degrees. a) Low parallelism degree, b) Medium parallelism degree, c) High parallelism degree.

degree $n/3 \le h < 2n/3$, and 3) task graphs with low parallelism degree is $2n/3 \le h < n$.

We also considered a different number of tasks in our experiments and conducted experiments on a platform with 16 cores, including 8 high-performance and 8 low-power cores. We compared TASS with two selected schemes (TP3M and ConvSS) for: *i*) the worst-case scenario when the system consumes the maximum possible power (Section 5.1) and *ii*) the realistic scenario, including both faulty and fault-free scenarios when the system consumes real power (Section 5.2).

### 5.1 The Worst-Case Scenario

In the worst-case scenario, the maximum power consumption by the system consumes because the main and backup copies of each task are executed in this scenario. Therefore, it can be considered a condition in which we compare feasibility, maximum temperature, QoS, and peak and average power consumption of TASS with the state-of-the-art methods. In order to analyze the feasibility, QoS, and peak power consumption for each system configuration, we used more task sets, and then the average results are reported. Each case was simulated 100 times with different

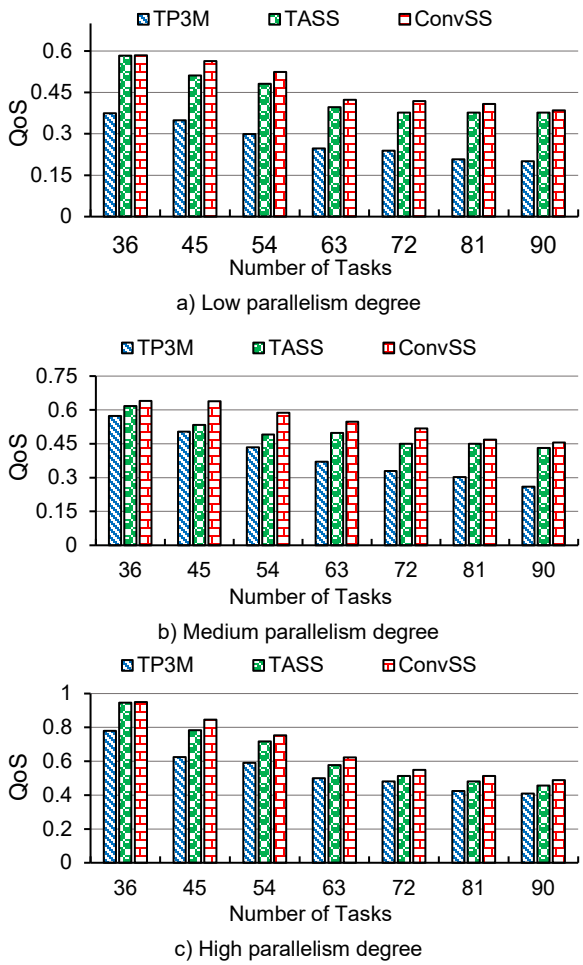


Fig. 6. Quality of service of schemes in the worst-case scenario for different parallelism degrees. a) Low parallelism degree, b) Medium parallelism degree, c) High parallelism degree.

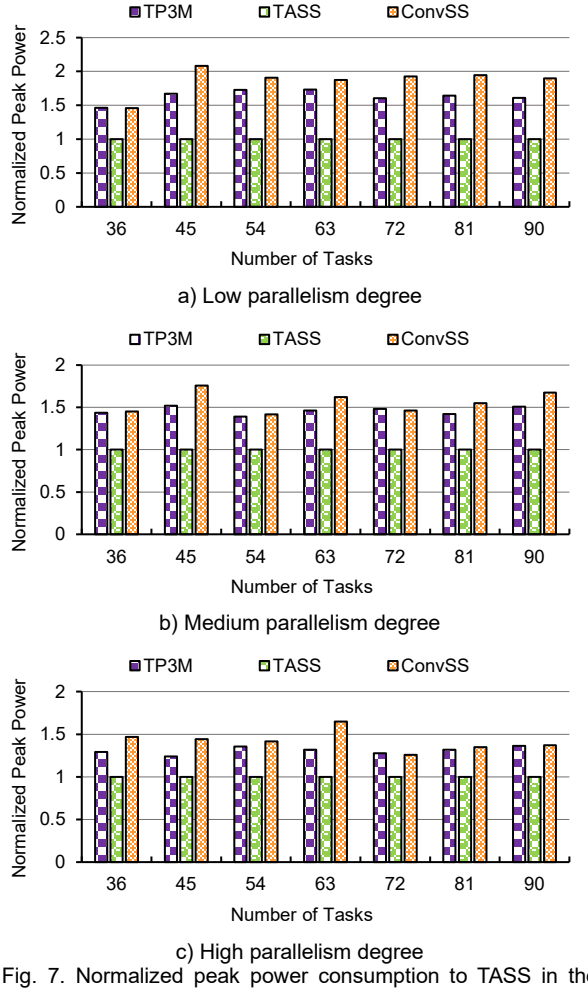


Fig. 7. Normalized peak power consumption to TASS in the worst-case scenario for different parallelism degrees. a) Low parallelism degree, b) Medium parallelism degree, c) High parallelism degree.

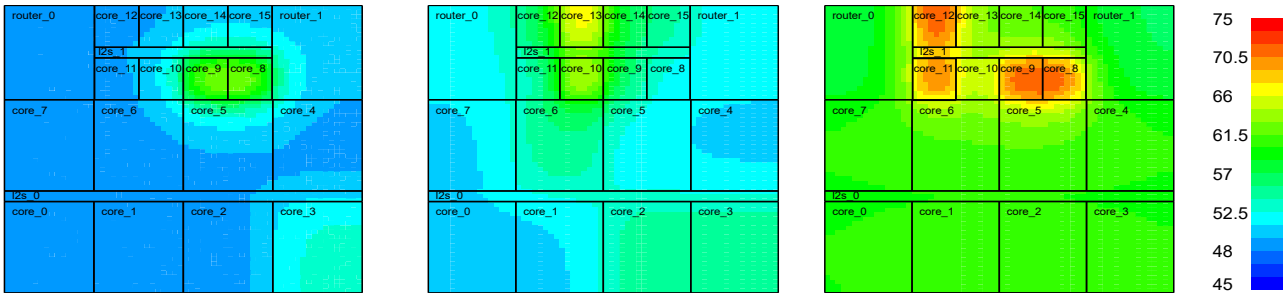


Fig. 8. The average transient temperature of a 16-core system for all methods by considering $T_{threshold}$= 60℃. a) TASS, b) TP3M, c) ConvSS.

parameters of the applications (i.e., tasks' worst-case and application deadline), and the average results are reported.

### 5.1.1 Evaluating Feasibility

We define feasibility as the percentage of satisfying both timing and power constraints in the resulting schedule. In Fig. 5, we show our comparison with two state-of-the-art techniques in terms of feasibility with different parallelism of the graph and the different number of tasks. The results consider the worst-case scenario where the main and backup copies of tasks will be executed completely. The parallelism degree is increased from Fig. 5a to Fig. 5c. Although a high parallelism degree implies higher feasibility, the power consumption will increase, and more thermal violations occurred, and therefore it can be observed how the state-of-the-art methods suffer from low feasibility.

### 5.1.2 Evaluating QoS

Fig. 6 represents the QoS of TASS in the worst-case scenario. In this evaluation, task-graphs with different parallelism degrees are considered, and evaluations are repeated for a different number of tasks. The amount of QoS of the worst-case scenario is computed based on the linear utility function (see Eq. 9). By changing the degree of the graphs, the QoS improvement is evaluated. In Fig. 6a to Fig. 6c, by increasing the degree of the graphs, the QoS improvement is increased because the tasks can execute in parallel. Note that the results of Fig. 6 are calculated based on Eq. 8 and have not the unit because in Eq. 8 we have divided the utility function which is a constant value and has not the unit (see Eq. 9) to the maximum number of tasks. Also, Eq. 9 is shown the linear utility function that has not the unit.

### 5.1.3 Evaluating Peak Power Consumption

In order to provide a more detailed analysis of peak power consumption, for each system configuration, we used more task sets, and then the average results of peak power consumption are reported in Fig. 7. This figure shows the normalized peak power consumption of different schemes to TASS in the worst-case scenario for different parallelism degrees. When the number of tasks increases, the peak power reduction of TASS is higher than other schemes. Also, when the parallelism degree of task graphs increases, the peak power consumption of TASS does not increase, while other schemes increase it. The peak power consumption of TASS is always less than the other two schemes. The results of peak power consumption show that TASS provides up to 51.94% (on average by 34.06%) peak power

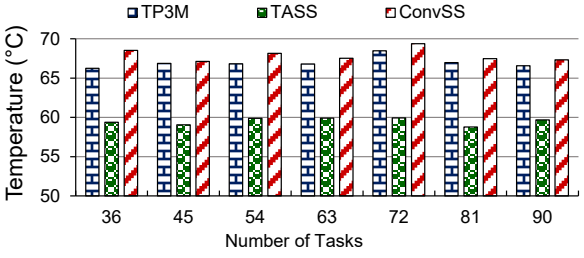


Fig. 9. The average results for peak temperature on the chip.

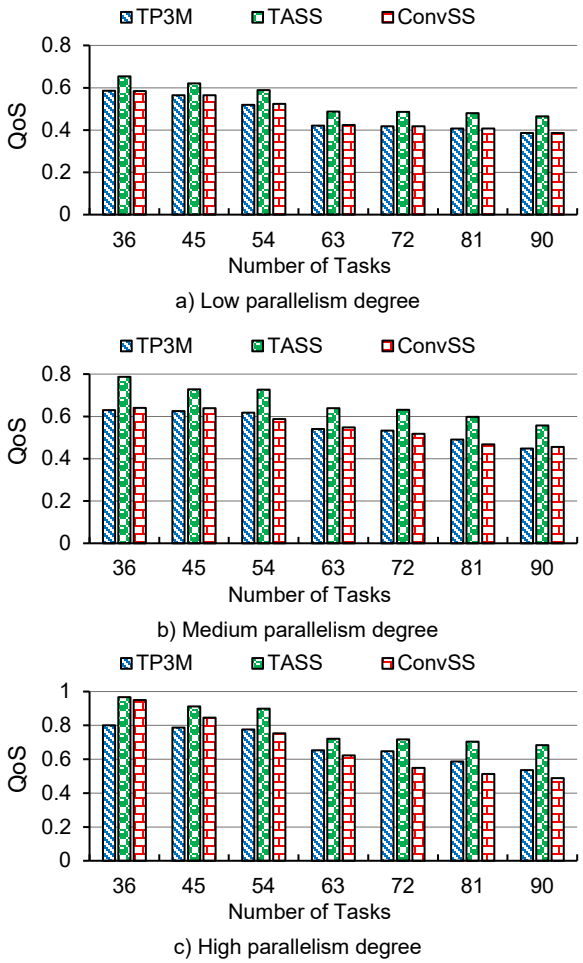


Fig. 10. Quality of service of schemes in the realistic scenario for different parallelism degrees. a) Low parallelism degree, b) Medium parallelism degree, c) High parallelism degree

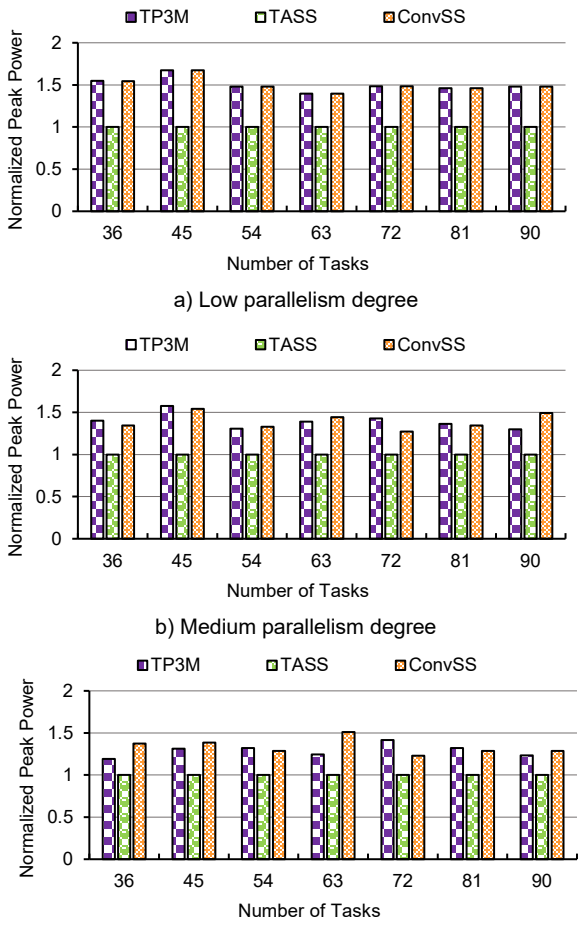


Fig. 11. Normalized peak power consumption to TASS in the realistic scenario for different parallelism degrees. a) Low parallelism degree, b) Medium parallelism degree, c) High parallelism degree.

reduction compared to state-of-the-art schemes.

### 5.1.4 Evaluating Temperature

In this subsection, we evaluate the resulting maximum temperature after applying the mapping and scheduling of the methods for different numbers of cores. Fig. 8 shows the average transient temperature of a 16-core system for all methods. In Fig. 8, the maximum temperature of TASS among all cores reaches about 59.96˚C, but other methods violate the temperature threshold, i.e., $T_{threshold}$= 60˚C. For better observation, we have reported the average result of different methods when we have again simulated 1000 times with different parameters of the applications (see Fig. 9). As shown in Fig. 9, the TASS method meets the temperature threshold in all configurations. Note that TASS meets the temperature constraint, but other methods do not consider temperature constraint and miss the temperature constraint. As it is shown in Fig. 9, in a higher number of cores, the maximum temperature increases because the number of active cores is increased. Our TASS method achieves up to 9.45˚C reduction in the system temperature compared to state-of-the-art methods while keeping the system reliability at a required level.

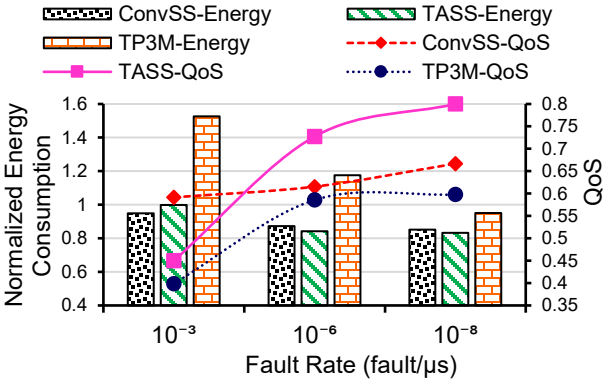


Fig. 12. The results of QoS and normalized energy consumption at runtime for different fault rates.

## 5.2 The Realistic Scenario

In this scenario, we investigate the realistic conditions where the actual fault injection is considered. In order to generate fault rate and pattern, transient faults are injected using a Poisson process where the fault rate corresponding to different voltage levels was generated using Eq. 4 under the parameters $\lambda_0$=$10^{-6}$ faults/μs and $d$ = 2. Therefore, the fault rate varies between $10^{-6}$ faults/μs corresponding to $V_{max}$ and $10^{-2}$ faults/μs corresponding to $V_{min}$. We generate a fault vector that determines at which times faults occur. Based on the generated fault vector, we decide that the task becomes faulty during the execution of a task set. Since transient faults are rare in nature, TASS achieves further temperature reduction in this scenario beyond what is achieved through the worst-case scenario. In this scenario, when the main task $T_i$ is executed successfully, its corresponding backup task is dropped from the schedule, and the slack time is released that can be exploited to further reduce the temperature. Fig. 10 represents the QoS of TASS in a realistic scenario. In this evaluation, similar to the worst-case scenario, task-graphs with different parallelism degrees are considered, and simulations are repeated for the different number of tasks. When the degree of the graphs increases, the QoS improvement is evaluated. In Fig. 10a to Fig. 10c, by increasing the degree of the graphs, the QoS improvement is increased because the tasks can be executed in parallel. Experimental results show that TASS meets the power constraints while, at the same time, improving the QoS of tasks, with an average of 18.40% (up to 39.78%). Fig. 11 shows the normalized peak power consumption of different schemes to TASS in the realistic scenario for different parallelism degrees. The results show

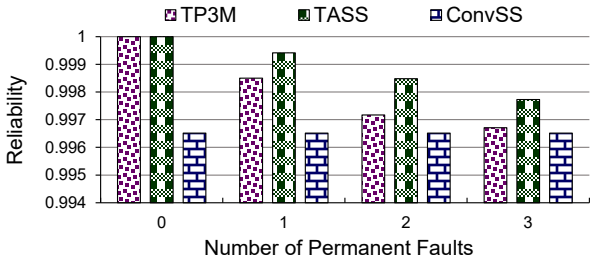


Fig. 13. The results of reliability at runtime for different number of permanent faults and the transient fault-rate modeled using Eq 4.

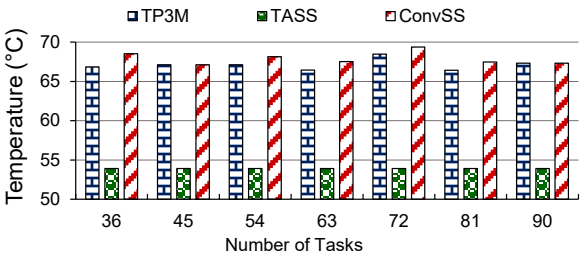


Fig. 14. The result of peak temperature in runtime for different methods.

that TASS provides up to 40.21% (on average by 28.31%) peak power reduction compared to state-of-the-art schemes.

It should be noted that our assumed fault rate is much higher than real fault rates (i.e., $10^{-12}$ faults/us based on [23]). Note that our assumption leads to lower QoS values. If we consider the real fault rate, we can achieve a higher QoS than our represented one. This is because using real fault rate will require much more number of fault injections to cover different parts of applications like other works in the literature [2][6][7][8][9][24][32], which will require months of simulations to ensure high coverage, and hence, we used the high fault rates to evaluate our scheme. Moreover, if we consider a higher fault rate than our assumed (e.g, $10^{-3}$ faults/us), we can see that the QoS decreases and the energy consumption increases because most of the backup tasks should be executed and we cannot drop them for further power and energy reduction. In order to verify this discussion, we have repeated our simulations for three different faults rates, i.e., 1) $10^{-3}$ faults/us, 2) $10^{-6}$ faults/us, and 3) $10^{-8}$ faults/us. As it can be seen from Fig. 12, the above discussion is approved.

In order to analyze the reliability of TASS, the transient fault-rate was modeled using Eq. 4 under parameters $\lambda_0$=$10^{-6}$ faults/us and $d$=2 [2]. Also, we consider that the permanent faults happen on the primary cores. Fig. 13 reports the reliability of applications for ConvSS, TP3M with N=3, and TASS. As seen in this figure, by increasing the number of permanent faults on the primary cores, the reliability of TASS is much higher than other schemes. Obviously, TP3M fails in tolerating permanent faults at different fault-rates, since it does not support tolerating the permanent faults.

In the realistic scenario, due to task dropping, the maximum temperature decreases. As it is shown in Fig. 14, our TASS method achieves up to 15.47˚C reduction in the system temperature, more than what has been achieved in the worst-case scenario compared to state-of-the-art methods while keeping the system reliability at a required level.

For the final discussion, if we exploit homogeneous multicore systems, we will face two situations: 1) if we consider that all of the cores are from the type of Low-Power cores, it may result in a lower QoS because the execution time of the main tasks increases; 2) if all of the cores are from the type of High-Performance cores, the peak power consumption and temperature increase. In two cases, when the main task finishes successfully, the whole of its corresponding backup task can be dropped, resulting in a significant amount of power and temperature reduction. Since both of them have some negative effects, we exploited a heterogeneous multicore system as a tradeoff to improve our goals.

## 6 Conclusions

In this paper, we have proposed a thermal-aware standby-sparing system where the main tasks and the backup tasks are scheduled on the primary and spare cores, respectively, such that the Thermal Safe Power (TSP) and real-time constraints are met. Our proposed method provides an opportunity to remove the overlaps of the execution of main and backup tasks to reduce temperature due to applying the fault-tolerant technique in fault-free scenarios. We have evaluated our proposed method under various system configurations and workloads. Our experiments show that TASS improves QoS up to 39.78% (on average by 18.40%) and reduces the peak power consumption and temperature by up to 40.21% and 15.47˚C (on average 28.31% and 13.60˚C), respectively, at runtime, while keeping the system reliability at the required level in realistic scenarios. Also, in the worst-case fault scenario, by considering the available slack times, our proposed method achieves, on average, 34.07% and up to 51.94% peak power improvement in comparison to the state-of-the-art methods.

## References

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

[2] 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," in *IEEE Transactions on Parallel and Distributed Sys.*, vol. 30, no. 1, pp. 161-173, 1 Jan. 2019.

[3] N. Goulding-Hotta, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, P. Huang, M. Arora, S. Nath, V. Bhatt, J. Babb, S. Swanson, and M. Taylor,"The greendroid mobile application processor: An architecture for silicon's dark future," *IEEE Micro*, vol. 31, no. 2, pp. 86–95, Mar./Apr. 2011.

[4] S. Pagani, H. Khdr, J. J. Chen, M. Shafique, M. Li and J. Henkel, "Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon," *IEEE Trans. on Comp.*, vol. 66, no. 1, pp. 147-162, 2017.

[5] H. Khdr, S. Pagani, E. Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel, "Power Density-Aware Resource Management for Heterogeneous Tiled Multicores," *IEEE Trans. on Computers*, vol. 66, no. 3, pp. 488-501, 1 March 2017.

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

[7] M. A. Haque, H. Aydin and D. Zhu, "On Reliability Management of Energy-Aware Real-Time Systems Through Task Replication," *IEEE Trans. on Par. and Dis. Sys.*, vol. 28, no. 3, pp. 813-825, 2017.

[8] M.A. Haque, H. Aydin, and D. Zhu, "Energy-Aware Standby-Sparing Technique for Periodic Real-Time Applications," *Proc. IEEE 29th Int'l Conf. Comput. Design (ICCD'11)*, pp. 190-197, Oct. 2011.

[9] 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," *IEEE Trans. on Par. and Dis. Sys.*, vol. 30, no. 10, pp. 2338-2354, 1 Oct. 2019.

[10] M. Ansari, J. Saberlatibari, S. M. Pasandideh and A. Ejlali, "Simultaneous Management of Peak-Power and Reliability in Heterogeneous Multicore Embedded Systems," *IEEE Trans. on Par. and*

*Dis. Sys.*, vol. 31, no. 3, pp. 623-633, 1 March 2020.

[11] C. LaFrieda, E. Ipek, J. F. Martinez and R. Manohar, "Utilizing Dynamically Coupled Cores to Form a Resilient Chip Multiprocessor," *37th Annual IEEE/IFIP International Conference on Dependable Sys. and Networks (DSN'07)*, Edinburgh, pp. 317-326, 2007.

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

[13] X. Zhu, X. Qin and M. Qiu, "QoS-Aware Fault-Tolerant Scheduling for Real-Time Tasks on Heterogeneous Clusters," *IEEE Transactions on Computers*, vol. 60, no. 6, pp. 800-812, June 2011.

[14] G. Buttazzo, "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications," New York, *NY: Springer*, 2011.

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

[16] 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," *42nd Annual IEEE/ACM Int. Symp. on MICRO*, pp. 469-480, 2009.

[17] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE TVLSI*, vol. 14, no. 5, pp. 501–513, 2006.

[18] 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," *Proc. 4th IEEE Ann. Workshop Workload Characterization*, 2001, pp. 3–14.

[19] D. Pradhan, Fault-tolerant computer system design. Upper Saddle River, *N.J.: Prentice Hall PTR*, 1996.

[20] S. Safari, S. Hessabi, and G. Ershadi, "LESS-MICS: A Low Energy Standby-Sparing Scheme for Mixed-Criticality Systems," *IEEE Trans. on Comp.-Aid. Des. of Integ. Circ. and Sys.*, 2020.

[21] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, "System-Level Design Techniques for Energy-Efficient Embedded Systems," vol. 53, no. 9. *Springer Science & Business Media,* 2004.

[22] T. Tobita, H. Kasahara, "A Standard Task Graph Set for Fair Evaluation of Multiprocessor Scheduling Algorithms," *J. Scheduling*, vol. 5, no. 5, pp. 379-394, 2002.

[23] A. Ejlali, B.M. Al-Hashimi, and P. Eles, "A Standby-Sparing Technique with Low Energy-Overhead for Fault-Tolerant Hard Real-Time System," *Proc. International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS)*, pp. 193-202, Grenoble, France, October 2009.

[24] M. A. Haque, H. Aydin, D. Zhu, "Energy-aware standby-sparing for fixed-priority real-time task sets," *Sustainable Computing: Informatics and Systems*, 2015, v. 6, pp. 81-93.

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

[26] M. A. Haque, H. Aydin, and D. Zhu, "Energy-aware Standby-Sparing Technique for periodic real-time applications," *2011 IEEE 29th International Conference on Computer Design (ICCD)*, Amherst, MA, 2011, pp. 190-197.

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

[28] 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," *Journal of Systems Architecture*, 2017, v 78, pp. 68-80.

[29] Y. Guo, D. Zhu, and H. Aydin, "Generalized Standby-Sparing techniques for energy-efficient fault tolerance in multiprocessor real-time systems," *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, Taipei, 2013, pp. 62-71.

[30] M. K. Tavana, M. Salehi, and A. Ejlali, "Feedback-Based Energy Management in a Standby-Sparing Scheme for Hard Real-Time Systems," *IEEE 32nd Real-Time Systems Symposium*, Vienna, 2011, pp. 349-356.

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

[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] G. J. Briggs, E. J. Tan, N. A. Nelson, and D. H. Albonesi, "QUILT: a GUI-based integrated circuit floorplanning environment for computer architecture research and education," *Proc. of Workshop on Computer Architecture Education*, 2005.

[34] Guocong Song and Ye Li, "Cross-layer optimization for OFDM wireless networks-part II: algorithm development," *IEEE Transactions on Wireless Communications*, vol. 4, no. 2, pp. 625-634, March 2005.

[35] 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," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[36] 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," *2019 IEEE Real-Time Systems Symposium (RTSS)*, Hong Kong, Hong Kong, 2019.

[37] H. Aydin, V. Devadas and D. Zhu, "System-Level Energy Management for Periodic Real-Time Tasks," *2006 27th IEEE Int. Real-Time Systems Symposium (RTSS'06)*, Rio de Janeiro, 2006.

[38] 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.

[39] S. Rehman, F. Kriebel, Duo Sun, M. Shafique, and J. Henkel, "dTune: Leveraging reliable code generation for adaptive dependability tuning under process variation and aging-induced effects," *51st ACM/EDAC/IEEE Design Automation Conf. (DAC)*, San Francisco, CA, 2014, pp. 1-6.

[40] J. Huang, K. Huang, A. Raabe, C. Buckl and A. Knoll, "Towards fault-tolerant embedded systems with imperfect fault detection," *Design Automation Conf. (DAC)*, San Francisco, CA, 2012, pp. 188-196.

[41] G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, and K. Li, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems," in *IEEE Trans. on Sus. Comp. (TC)*, vol. 3, no. 3, pp. 167-181, 1 July-Sept. 2018.

[42] M. Salehi, M. Khavari Tavana, S. Rehman, F. Kriebel, M. Shafique, A. Ejlali, and J. Henkel, "DRVS: Power-efficient reliability management through dynamic redundancy and voltage scaling under variations," *IEEE/ACM Int'l Symposium on Low Power Electronics and Design (ISLPED)*, Rome, 2015, pp. 225-230.

**Mohsen Ansari** received the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2016. He is currently working toward the Ph.D. degree in computer engineering at Sharif University, Tehran, Iran, from Sept. 2016 until now. He is now a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany, from Oct. 2019 until now. Also, he is a group leader of the Embedded Systems Research Laboratory (ESR-LAB) at the department of computer engineering, Sharif University of Technology. His

research interests include the low-power design of embedded systems and multi-/many-core systems with a focus on dependability/reliability.

**Sepideh Safari** received the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2016. She is currently working toward the Ph.D. degree in computer engineering at the Sharif University of Technology. She is now a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany. Her research interests include the low-power design of cyber-physical systems, energy management in fault-tolerant embedded systems, and multi-/many-core systems with a focus on dependability/reliability.

**Sina Yari-Karin** received his B.Sc. degree in computer engineering from the Ferdowsi University of Mashhad in 2017. He is currently an M.Sc. student in computer engineering at the Sharif University of Technology, Tehran, Iran. Also, he is a member of the Embedded Systems Research Laboratory (ESR-LAB) at the department of computer engineering at the Sharif University of Technology. His research interests are embedded system design, low power system design, fault-tolerant system design, and computer architecture.

**Pourya Gohari-Nazari** received the B.Sc. degree in computer engineering from the University of Isfahan. He is currently working toward the M.Sc. degree in the Department of Computer Engineering at the Sharif University of Technology, Tehran, Iran. His research interests are thermal management in many-core systems and design embedded systems with a focus on low-power and reliability.

**Heba Khdr** is a postdoctoral researcher and a group leader at the Chair for Embedded Systems (CES) in Karlsruhe Institute of Technology (KIT) in Germany. She received her Ph.D. (Dr.-Ing.) in Computer Science from Karlsruhe Institute of Technology (KIT) in 2018.
In 2005, she received her Diploma in Informatics Engineering from Aleppo University in Syria with an excellent grade and the first rank.
From 2005 until 2007, she worked as a software engineer in the industry sector in Syria. She worked as an assistant at Aleppo University from 2008 until 2010. In 2011 she did an equivalent master thesis at KIT. Her research interests are thermal management and resource management in multi- and many-core systems. In 2012 she received Research Student Award from KIT. She received Best Paper Award from IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) in 2014 and four HiPEAC paper awards.

**Muhammad Shafique** (M'11-SM'16) received his Ph.D. in computer science from Karlsruhe Institute of Technology, Germany, in 2011. From Oct.2016 to Aug.2020, he was a full professor at the Institute of Computer Engineering, TU Wien, Austria. Since Sep.2020, he is with the Division of Engineering at New York University Abu Dhabi (NYUAD) and is a Global Network Faculty at the NYU Tandon School of Engineering, USA. His research interests are in system-level design for brain-inspired computing, AI/Machine Learning hardware, wearables, autonomous systems, energy-efficient and robust computing, IoT, and Smart CPS. Dr. Shafique has given several Keynotes, Talks, and Tutorials and organized special sessions at premier venues. He has served as the PC Chair, General Chair, Track Chair, and PC member for several conferences. He received the 2015 ACM/SIGDA Outstanding New Faculty Award, AI 2000 Chip Technology Most Influential Scholar Award in 2020, six gold medals, and several best paper awards and nominations.

**Jörg Henkel** (M'95-SM'01-F'15) received the Diploma and Ph.D. (summa cum laude) degrees from the Technical University of Braunschweig, Germany. He was a Research Staff Member with NEC Laboratories, Princeton, NJ, USA. His research work is focused on co-design for embedded hardware/software systems with respect to power, thermal, and reliability aspects. Dr. Henkel has received six best paper awards from, among others, ICCAD, ESWeek, and DATE. He served as the Editor-in-Chief for the ACM TECS and IEEE Design&Test. He is/has been an Associate Editor for major ACM and IEEE journals. He was a General Chair ICCAD, ESWeek, etc., and serves as a Steering Committee chair/member for leading conferences and journals. He coordinates the DFG Program SPP 1500 "Dependable Embedded Systems" and is a site coordinator of the DFG-TR89 collaborative research center on "Invasive Computing." He is the Chairman of the IEEE Computer Society, Germany Chapter, and a Fellow of the IEEE.

**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. In 2006, he joined the Sharif University of Technology as a faculty member in the department of computer engineering, and from 2011 to 2015, he was the director of the Computer Architecture Group in this department. He is now the director of Embedded Systems Research Laboratory (ESR-LAB) and the head of the department of computer science and engineering, Sharif University of Technology. His research interests include low power design, fault tolerance, real-time embedded systems, and Internet of Things (IoT).