IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

A Survey of Fault-Tolerance Techniques for Embedded Systems from the Perspective of Power, Energy, and Thermal Issues

SEPIDEH SAFARI[1,2], MOHSEN ANSARI[1,2], HEBA KHDR[2], POURYA GOHARI NAZARI[1], SINA YARI-KARIN[1], AMIR YEGANEH-KHAKSAR[1], SHAAHIN HESSABI[1], (Member, IEEE), ALIREZA EJLALI[1], AND JÖRG HENKEL[2], (Fellow, IEEE)

[1]Department of Computer Science and Engineering, Sharif University of Technology, Tehran 14588, Iran
[2]Department of Computer Science, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany

Corresponding authors: Shaahin Hessabi (hessabi@sharif.edu).

ABSTRACT The relentless technology scaling has provided a significant increase in processor performance, but on the other hand, it has led to adverse impacts on system reliability. In particular, technology scaling increases the processor susceptibility to radiation-induced transient faults. Moreover, technology scaling with the discontinuation of Dennard scaling increases the power densities, thereby temperatures, on the chip. High temperature, in turn, accelerates transistor aging mechanisms, which may ultimately lead to permanent faults on the chip. To assure a reliable system operation, despite these potential reliability concerns, fault-tolerance techniques have emerged. Specifically, fault-tolerance techniques employ some kind of redundancies to satisfy specific reliability requirements. However, the integration of fault-tolerance techniques into real-time embedded systems complicates preserving timing constraints. As a remedy, many task mapping/scheduling policies have been proposed to consider the integration of fault-tolerance techniques and enforce both timing and reliability guarantees for real-time embedded systems. More advanced techniques aim additionally at minimizing power and energy while at the same time satisfying timing and reliability constraints. Recently, some scheduling techniques have started to tackle a new challenge, which is the temperature increase induced by employing fault-tolerance techniques. These emerging techniques aim at satisfying temperature constraints besides timing and reliability constraints. This paper provides an in-depth survey of the emerging research efforts that exploit fault-tolerance techniques while considering timing, power/energy, and temperature from the real-time embedded systems' design perspective. In particular, the task mapping/scheduling policies for fault-tolerance real-time embedded systems are reviewed and classified according to their considered goals and constraints. Moreover, the employed fault-tolerance techniques, application models, and hardware models are considered as additional dimensions of the presented classification. Lastly, this survey gives deep insights into the main achievements and shortcomings of the existing approaches and highlights the most promising ones.

INDEX TERMS Fault-tolerance, embedded systems, real-time computing, scheduling, power/energy minimization, thermal-aware design.

I. INTRODUCTION

Aggressive scaling in the size of the transistors enables integrating billions of transistors into a single die, which significantly improves computation performance [1]. Nevertheless, technology scaling has led to several negative impacts on system reliability [2][3]. Firstly, it increases the rate of radiation-induced faults up to several orders of magnitude [4][5][81]. Secondly, it increases the power density on the chip, and thereby on-chip temperatures are elevated [6]. High temperature, poses considerable challenges to lifetime reliability (occurrence of permanent faults) due to their direct influence on the aging effects such as electro migration. Moreover, it may increase soft error rates [7]. In order to mitigate thermal violation chip-level

countermeasures such as lowering the operating voltage and frequency will be triggered, which can again significantly affect the timeliness and reliability of the system. Hence, system reliability has become a major concern in real-time embedded systems design, due to the negative impacts of technology scaling. Automotive systems, avionics, satellite, robots, and wireless body area networks (WBAN) are the example of embedded systems which have been growing steadily in the recent past and should meet the correctness and timeliness even under fault occurrences [10][13][14][15].

To ensure reliability in embedded systems, so-called *fault-tolerance techniques* have emerged [16][17]. Particularly, they employ redundancy in terms of time, hardware, software, and information to satisfy a given reliability target, which specifies the probability that the system functions correctly according to its specifications in the time interval [0, t], with the assumption that it was functioning correctly in the beginning (at time 0) [16][17]. The shift to a multi-core paradigm provides a great potential for the implementation of fault-tolerance techniques, which require additional resources on the chip to employ redundancy in order to fulfill reliability requirements [83][92]. However, implementing fault-tolerance techniques leads to time overhead, additional power/energy consumption, and high temperature.

In general, to suppress power and temperature on the chip, several countermeasures such as Dynamic Power Management (DPM) and Dynamic Voltage and Frequency Scaling (DVFS) can be taken by the dynamic thermal management (DTM) unit that is typically implemented on the chip [19]. However, DVFS can potentially degrade the system reliability because the rate of transient faults increases at low supply voltages [20][62][66]. Moreover, such countermeasures might prevent the tasks from meeting their deadlines, which is not acceptable in real-time embedded systems [21]. That implies the goals/constraints of designing real-time embedded systems, i.e., low-power consumption, real-time computing, and high reliability are contradicting each other as depicted in Figure 1. Therefore, it is indispensable to consider all of those metrics; i.e., time, power, and temperature, within the task mapping and scheduling policies of the fault-tolerance systems in order to find a suitable trade-off and avoid triggering conservative countermeasures.

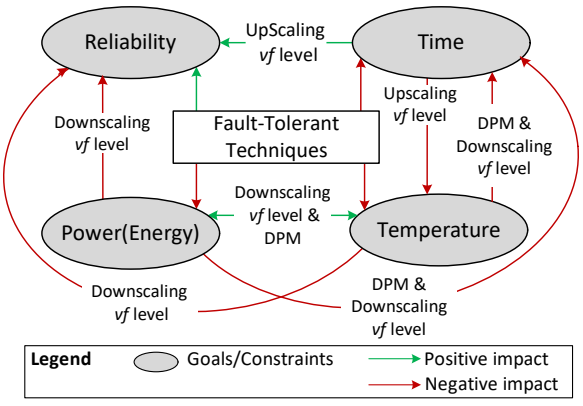

FIGURE 1. Illustrating the dependencies between the contradictory constraints and goals of fault-tolerance techniques.

Besides existing classification in the state-of-the-art survey papers, we consider an additional dimension which is the considered goals and constraints of the mapping and scheduling policies in fault-tolerance embedded systems (i.e., time, power/energy, and temperature). This survey shows that the majority of the state-of-the-art fault-tolerance techniques either solely focus on satisfying timing constraints or optimizing for power/energy as well. Just a few state-of-the-art fault-tolerance techniques consider the thermal issue, despite its relevance. Particularly, employing fault-tolerance techniques in embedded systems leads to increase the temperature, and therefore temperature constraints need to be considered. Hence, our survey paper shows the significant impact of employing fault-tolerance techniques on the temperature through a motivational example. Then, it summarizes and classifies the state-of-the-art fault-tolerance techniques considering the taxonomy presented in Figure 2. Additionally, this survey highlights the advantages and the shortcomings of state-of-the-art techniques.

The remainder of this paper is organized as follows. Section II discusses the system model, and fault-tolerance techniques. Section III surveys the studies that exploit fault-tolerance techniques without considering power, energy, and temperature constraints. Section IV surveys the power/energy-aware fault-tolerance techniques. Thermal-aware fault-tolerance techniques are studied in Section V. Finally, the paper is concluded in Section VI.

II. System Models and Fault-Tolerance Techniques

The design of fault-tolerance techniques depends on the targeted system model, which is described by software-level and hardware-level parameters. In particular, selecting the redundancy type of fault-tolerance techniques depends to a large extent on the system model. Moreover, mapping and scheduling techniques will be designed based on both the

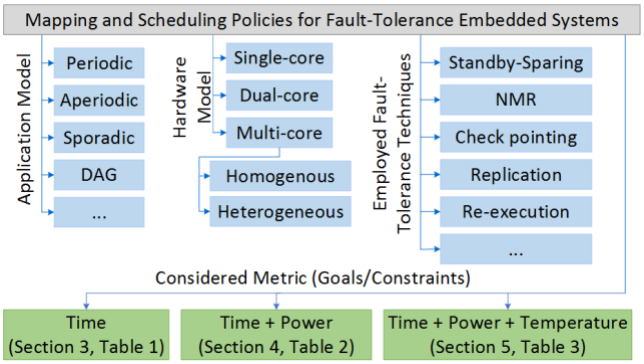

FIGURE 2. Classification of system-level fault-tolerance techniques in real-time embedded systems.

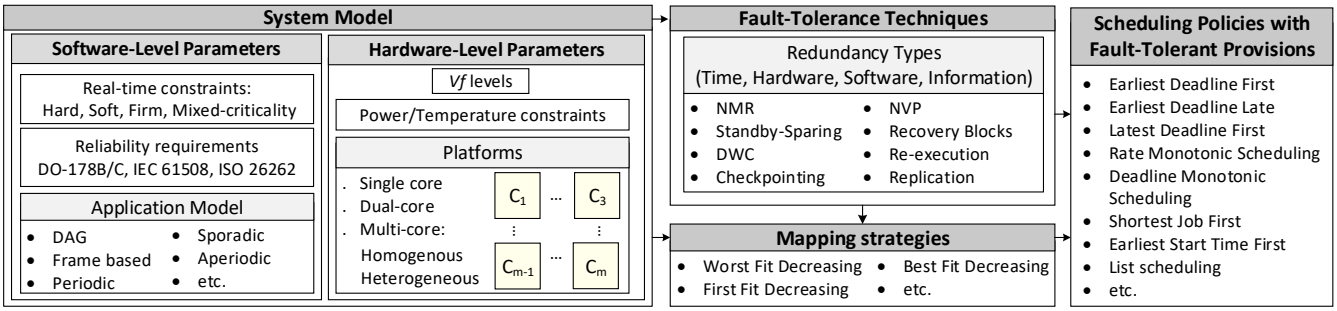

FIGURE 3. Overview of fault-tolerance systems.

employed fault-tolerance techniques and the targeted system model, while considering the target goals and constraints of the system. Figure 3 summarizes the available options for system models, fault-tolerance techniques, and mapping/scheduling policies.

A. System Model

1) SOFTWARE-LEVEL PARAMETERS

The main software-level parameters include the real-time constraints, reliability requirements, and application model as shown in Figure 3.

Real-time constraints: Embedded systems are information processing systems that are embedded into a larger product. Meeting real-time constraints, dependability, energy efficiency, code-size efficiency, low weight, low cost, and run time efficiency are common characteristics of these systems. Indeed, real-time embedded systems are computing systems that react to environmental events within precise time constraints [10]. Therefore, the correct output of these systems depends on both the correct result and the time at which the results are generated [10]. Hence, based on the consequences that may happen after missing the predefined timing constraint, real-time tasks are categorized into the following groups [10]:

- Hard: If producing the results after missing the deadline leads to catastrophic consequences, a real-time task is known as hard [10].
- Firm: If producing the results after missing the deadline is useless for the system, but does not cause any damage, a real-time task is known as a firm [10].
- Soft: If producing the results after missing the deadline has still some utility for the system, although causing performance degradation, a real-time task is known as soft [10].

Recently advancement of cyber-physical systems attracted more attention to mixed-criticality systems (MCSs). In MCSs a large number of tasks of different criticality levels, with different timing requirements, are integrated to execute on the same computing platform, to meet stringent non-functional requirements relating to the area, cost, and power [11].

Reliability requirements: The reliability of the system specifies the probability that the system functions correctly according to its specifications in the time interval [0, t], with the assumption that it was functioning correctly in the beginning (at time 0) [16][17]. The reliability requirements of the system will be determined based on different safety standards such as IEC61508 for all kinds of industrial software systems, DO-254, ISO26262 for automotive systems, and DO-178B/C for avionic systems [22]. More details about the different types of fault occurrence on the system and fault-tolerance techniques are described in Section II.C.

Application Model: There are several application models executing on a real-time system. Computational activities can be done independently or dependent on each other. In some applications, computational activities have to respect some precedence relations defined at the design time. Such precedence relations are usually described through a directed acyclic graph, where tasks are represented by nodes and precedence relations by arrows. Based on the periodicity of the execution of tasks (task activation), they can be periodic, aperiodic, or sporadic. The task model is called periodic when each task consists of infinite jobs, and jobs have regular inter-arrival time equal to the task's period. An aperiodic task has infinite jobs where jobs arriving at irregular intervals. A sporadic task is an aperiodic task where jobs have a minimum inter-arrival time.

2) HARDWARE-LEVEL PARAMETERS

The hardware architecture, i.e., single-core, dual-core, or multi-core processors[1] (including homogenous and heterogeneous), plays an important role in selecting the fault-tolerance technique, and the task mapping/scheduling policies. Moreover, available voltage and frequency levels (supporting discrete or continuous configuration), power/temperature constraints, and cooling system are other important architecture parameters.

B. Mapping and Scheduling Policies

After assigning tasks to cores (mapping) based on heuristic or optimal methods, tasks should be properly scheduled among cores to guarantee the constraints of the system. Proposed scheduling algorithms for real-time systems can be classified based on the following metrics:

- Preemptive or non-preemptive: In preemptive scheduling, an executing task will be interrupted at any

[1] The term multi-core is used in this paper to express both multi-core processor or multi-processors.

time, and the processor will be assigned to another task based on upcoming conditions. However, in non-preemptive scheduling, a task that starts its execution will not be interrupted until its completion.

- Static or dynamic: In static scheduling, the scheduling decision is fixed and assigned to tasks before system activation. While in dynamic scheduling decisions may be changed during system operation.
- Offline or online: in the offline scheduling, the proposed algorithm is applied in the offline phase (at design time) to the whole task set and the generated schedule is stored in a table to be exploited later. However, in online scheduling, every time a new task enters the system at runtime, the scheduling decisions are taken.
- Optimal or heuristic: The optimal schedule minimizes some given cost function defined over the task set. However, in heuristic-based algorithms, the scheduling decisions are taken based on proposed heuristic functions. Therefore, the heuristic algorithm tends toward the optimal schedule but does not guarantee to find it.

C. Fault-Tolerance Techniques

Faults in computer systems are classified into transient, intermittent, and permanent based on their occurrence and duration [16][17].

- Transient faults: This type of fault occurs for a short time period and then disappears without physical damage to the processor. It is often induced by electromagnetic interference and cosmic radiation.
- Intermittent faults: This type of fault occurs frequently, and it is difficult to detect because after its occurrence the system operates correctly.
- Permanent faults: This type of fault results from hardware component failure or manufacturing defects. Recovery from this kind of fault is only possible by replacing or repairing the faulty component.

In this survey paper, we targeted all types of faults in the state-of-the-art. Irrespective of the fault type, fault-tolerance techniques aim at detecting the faults and recover from them (if possible), to let the system continue to function correctly. Typically, a fault-tolerance technique is designed to satisfy a given reliability target. To do that, fault-tolerance techniques employ redundancy in terms of 1) hardware, 2) software, 3) information, and 4) time beyond what is needed for the normal operation of the system. In the following, we explain the different fault-tolerance techniques that belong to the four mentioned redundancy types as introduced in [16][17][24].

1) HARDWARE REDUNDANCY

Hardware redundancy is the most common technique which is the addition of extra hardware components for detecting or tolerating faults [16][17]. For example, instead of using a single core/processor, more cores/processors can be exploited, so that each application is executed on each core/processor, then the fault can be detected or even corrected. Hardware redundancy can be applied through passive, active, or hybrid methods.

Passive hardware redundancy: Examples of this redundancy are *N modular redundancy (NMR)* such as *Triple Modular Redundancy (TMR)*, and *voting* techniques. These techniques are referred to as M-of-N systems, which means that the system consists of *N* components, and the correct operation of this system is achieved when at least *M* components correctly work. The TMR system is a 2-of-3 system with M=2 and N=3, which is realized by three components performing the same action, and the result is voted [16][17].

Active hardware redundancy: *Duplication with comparison (DWC)*, *Standby-sparing (SS)*, *Pair-and-a-spare* technique, and *watchdog timers* are included in this type of active hardware redundancy. In DWC, two identical hardware components perform the same computation in parallel and their output is compared. Therefore, the DWC technique can only detect faults, but it cannot tolerate them because the faulty component cannot be determined [16][17]. In standby-sparing, one module is operational and one or more modules are standby or spares. If the fault is detected in the main component, it will be omitted from the operation and the spare component will continue the execution [16][17]. Meanwhile, pair-and-a-spare is a combination of DWC and SS techniques, i.e., two modules are executed in parallel and their results will be compared to detect the fault [16][17].

Hybrid hardware redundancy: The basic concept of this method is combining the features of both active and passive hardware redundancies. *N modular redundancy with spare*, *sift-out modular redundancy*, *self-purging redundancy*, and *triple duplex architecture* are examples of hybrid hardware redundancy [16][17]. The basic concept of self-purging is based on NMR with spare techniques, all modules are active and participate in the function of the system. In sift-out modular redundancy, there are *N* identical modules. However, they are configured in the system through special circuits (comparators, detectors, and collectors). The triple duplex architecture combines the DWC technique with TMR, which helps to detect the faulty module and remove it from the system.

2) TIME REDUNDANCY

Time redundancy is achieved by allocating extra time to perform the functions of the system to detect faults, and often tolerate them. It should be noted that applying time redundancy techniques must not lead to missing the timing constraints of real-time systems. For example, the *re-execution* technique is a well-known method of time redundancy that is the repetitive execution of the faulty task on the same hardware and comparing the results until reaching the correct output [16][17].

3) INFORMATION REDUNDANCY

Error detection and correction coding (such as *parity check*, *cyclic code*, *checksum*, etc.) are well-known information redundancy techniques [17]. Indeed, information redundancy is widely used in memory units, storage

devices, and data communication over noisy channels. *Redundant Arrays of Independent Disks (RAIDs)* are other well-known examples of information redundancy at a higher level than individual data words which have different organizations. Also, the *replication* technique is a well-known example of exploiting information redundancy to enhance reliability at the system level. In replication, identical copies of the data will be executed on multiple hardware.

4) SOFTWARE REDUNDANCY

Software redundancy is the addition of extra software to give an output for a desired function to detect and tolerate faults (if possible). *N version programming* (*NVP*), *checkpointing* (*CP*), and *recovery blocks* are well-known techniques that fall into the software redundancy category [16]. In *N* version programming, the software module is designed and coded *N* times by separate groups of programmers and the results are compared. Therefore, when *N* different programmers implement a specific software the likelihood of occurring the same mistake in all modules will be decreased. In the checkpointing technique, the last fault-free state of the faulty process is stored in advance in the stable memory. Whenever a fault occurs, the system rolls back to the last correct checkpoint and re-executes the application part that is executed in the last checkpoint duration [16][17].

It is worthy to mention that regarding existing redundancy types (hardware, software, information, and time) there is another classification known as spatial and temporal redundancies, in which temporal and spatial are analogous to hardware and time redundancy, respectively. However, information and software redundancies are a mix of spatial and temporal [23].

Fault-tolerance techniques can be classified into three broad categories based on exploited redundancy types including: *i*) hardware-based, *ii*) software-based, and *iii*) hybrid techniques [24].

Hardware-based techniques add extra hardware modules which changes the original architecture of the system or its components. Therefore, such techniques must be implemented during the design of the system. Hardware-based techniques have two main groups including redundancy-based, and hardware monitors. The first group relies on hardware or time redundancy, while the second group adds special hardware modules to the system's architecture to monitor the control flow of the programs inside the processors and memory accesses performed by them such as watchdog processors [25], checkers [26] or Infrastructure Intellectual Properties (I-IP) [27]. Hardware-based techniques have a high cost, verification and testing time, and area overhead which leads to higher power consumption as well.

Software-based techniques exploit the concepts of software, time, and information redundancies to detect faults during the execution of the program. Software-based techniques are divided into two groups: *i*) data flow checking techniques, which consider the faults in the data structures of the processor, such as variables, registers, and the data memory. These faults may lead to calculate an incorrect result, but they do not change the program flow. Error Detection by Data Diversity and Duplicated Instructions (ED^4I) [28], the transformation technique proposed in [29], and Variables 1 (VAR1), Variables 2 (VAR2), and Variables 3 (VAR3) [30] are the techniques that exploit information and software redundancies. *ii*) Control flow checking techniques, which deviate from the normal program flow and lead to an infinite loop in a subroutine or instruction. In the software-based techniques since there is no need to modify the hardware, they provide more flexibility, and low cost, and development time. However, the performance degradation is the main drawback, because extra instructions will be executed by the processor, which slows the overall application runtime, and increases the memory overhead.

Hybrid techniques are a combination of hardware-based and software-based techniques. Hybrid techniques have low development time and low area overheads (from the software-based techniques perspective), and low-performance degradation (from the hardware-based techniques perspective). However, they need the application source code, which is not always available, and require changes, at least, in the system's architecture. The studies in [27][32][33] are the example of hybrid fault-tolerance techniques in embedded systems. For example, the studies in [31][32] have proposed hardening infrastructure offers the techniques called SWIFT-R, as a software-based technique, and selective TMR as a hardware-based technique.

Fault-tolerance techniques can be applied at different levels of implementation, starting from the software level down to the architecture description level, the logical and transistor level, until the layout level. Regarding fault-tolerance techniques which are applied at a lower level such as Moore-Shannon's hammock networks which means that by replacing all transistors in a chip with hammock networks of transistors, the reliability will be enhanced while power consumption stays the same [34]-[37], in this survey paper, we have considered system-level fault-tolerance techniques, which have negative effects on the timing, power/energy, and temperature constraints of the system.

In the following, we present the three categories of the mapping/scheduling policies of fault-tolerance systems according to our classification (Figure 2). It is worthy to mention that in all categories timing constraints are considered as an important design metric. In the first classification (Section III), fault-tolerance techniques that only consider timing constraints as design metric is studied. In the second classification (Section IV), fault-tolerance techniques that consider power/energy management besides timing constraints are studied. All of the methods that are investigated in Section III and Section IV have ignored the effects of elevated temperature caused by the power density on the system. Therefore, thermal management to avoid

Table 1. Summary of timing-aware fault-tolerance techniques without considering power, energy, and temperature in real-time embedded systems.

Ref.	Application Model	Architecture Model	Fault-Tolerant Technique	Goals/Constraints
[38]	Periodic	Dual-core	Standby-Sparing (SS), Primary/Backup (P/B)	Timing/Reliability
[39][40]	Periodic	Homogeneous Multi-core		Timing/Reliability
[41]	Aperiodic	Homogeneous Multi-core		Timing/Reliability
[42][43]	Periodic	Homogeneous Multi-core		Timing/Reliability/Performance
[44]	Multi-DAG	Homogeneous Multi-core	Replication	Timing/Reliability
[45]	Periodic	Heterogeneous Multi-core		Timing/Reliability
[46][47]	Periodic	Homogeneous Multi-core		Timing/Reliability
[48]	Periodic/Sporadic	Single-core, Heterogeneous Multi-core	Checkpointing (CP)	Timing/Reliability
[49]	Single task	Single-core		Latency/Reliability
[50]	Periodic/Sporadic	Single-core		Timing/Reliability
[51]	Task-graph	Homogeneous Manycore	N Version Programming (NVP)	Reliability/Performance
[52]	Periodic	Homogeneous Multi-core		Tuning/ Reliability
[53]	Frame-based DAG	Single-core	Re-execution	Timing/Utility/Reliability
[54]	DAG	Single-core, Multi-core		Timing/Reliability
[55]	Periodic	Homogeneous Multi-core		Timing/Reliability/QoS/Utilization
[56][57]	Periodic/Sporadic	Single-core		Timing/Reliability
[58]	Sporadic	Single-core, Multi-core		Timing/Reliability/QoS
[59]	Frame-based	Single-core		Timing/Reliability
[60]	Multi DAG	Homogeneous Multi-core	Replication, Re-execution	Timing/Reliability
[61]	DAG	Heterogeneous Multi-core	Replication, Re-execution	Timing/Reliability
[62]	Periodic DAG	Heterogeneous Multi-core	Replication, CP	Timing/# checkpoints/Mapping
[63]	Sporadic	Single-core, Homo. Multi-core	DMR, TMR, Replication	Timing/Reliability/QoS

temperature-induced failures is also a significant research issue especially for real-time embedded systems with timing constraints and limited cooling techniques. As far as we know, little investigation has been conducted in the literature on thermal management for fault-tolerance real-time embedded systems. Therefore, finally, in the third classification (Section V) fault-tolerance techniques that consider temperature constraints besides timing constraints are studied. In all three classifications, state-of-the-arts are described based on exploited fault-tolerance techniques in detail including describing the exploited application model, architecture model, energy management techniques, goals, and constraints.

III. Timing-Aware Fault-Tolerance Techniques without Considering Power, Energy, and Temperature

The proposed mapping/scheduling policies in this category (summarized in Table 1) focus only on satisfying timing, and reliability constraints. However, exploiting fault-tolerance techniques will increase the power/energy and temperature which are not considered in the proposed methods in this category. Furthermore, we classify these techniques into sub-categories according to their employed fault-tolerance technique.

A. Standby-Sparing (SS) and Primary/Backup (P/B)

The problem of fixed-priority preemptive scheduling for a set of periodic hard real-time tasks has been proposed in [38], where each task has primary and backup versions. The primary version is more complex and has more functions that produce results with good quality, but its execution is more susceptible to faults because of its high level of complexity and resource usage. By contrast, the backup version is simpler and contains the minimum required functions, which produces acceptable results with lower precision. The proposed scheduling algorithm satisfies the timeliness of the primary and backup versions of each task, while attempts to complete as many primary tasks as possible. If the primary fails due to missing timing constraints or fault occurrence or when the latest time to start execution of the backup without missing the corresponding task deadline is reached, the backup will be executed. The experimental evaluations are based on simulation, and the real-world platform and application models are not used.

Kim et al. [39] have presented R-BATCH (Reliable Binpacking Algorithm for Tasks with Cold standby and Hot standby) scheme. Based on the required recovery time for tasks, it has considered hard recovery, soft recovery, and best-effort recovery tasks set. Then, it has introduced the idea of exploiting hot standby for hard recovery tasks, and cold standby for soft recovery and best-effort recovery tasks. They have proposed an allocation method, called Reliable Best-Fit Decreasing (R-BFD), for hot standby replicas which allocates active replicas such that the primary task and its corresponding active replicas are not assigned to the same processor. The proposed R-BATCH algorithm reduces the number of required replicas in comparison to R-BFD through considering cold standbys (passive replicas) which are activated whenever a fault occurs. The evaluation results demonstrate a significant improvement in the performance; however, the proposed method does not consider the impact of redundancy in the energy/power consumption of the

embedded system. Also, the evaluations are based on random task generation.

Kim et al. [40] have proposed a System-level Architecture for Failure Evasion in hard Real-time applications (SAFER) which observes the state and information of each task and whenever a failure is detected, using their fault detection method, SAFER reconfigures the system to maintain the functionality of the whole system. The SAFER proposed an architecture that can detect both time-based and event-based failures in a distributed embedded real-time system with periodic tasks. Moreover, the proposed configuration is implemented on Linux and x86 hardware. However, the proposed architecture does not consider the power consumption overhead of the redundant units, and it just evaluates the timing overhead.

A primary/backup online scheduling approach has been proposed in [41] that guarantees the reliability of the hard real-time system without increasing overhead and the need for extra hardware components. In the primary/backup approach, two identical versions for each task are scheduled on two different cores in a way they do not have any execution overlap. Hence, in the fault-free scenario, the overhead of the system is kept low; i.e., the backup version is executed whenever a fault is detected. Moreover, to efficiently map and schedule primary and backup versions on the cores, this paper has proposed two policies for homogenous multi-cores known as exhaustive search, and first found solution search (FFSS). All cores are checked through the exhaustive search for finding available slack times, then FFSS selects the best solution which schedules the primary version as soon as possible and the backup version as late as possible. However, FFSS determines the first suitable slot for the primary version and then for the first backup version without considering their positions within the schedule window. It has shown that FFSS reduces the computation complexity more than the exhaustive search. The scheduling approach in [41] with the FFSS policy achieves a significant improvement in the performance of periodic applications. However, the simulations do not consider real-world task sets and platforms. This proposed method attempts to execute primary tasks as soon as possible and the backup tasks as late as possible. Therefore, another missing subject in this proposed method is the impact of overlap minimization through the FFSS policy on the energy/power consumption of the system.

The authors in [42] have proposed two fault-tolerance techniques by presenting fixed-priority-based scheduling algorithms. The first technique (called Tercos) terminates the execution of backup tasks whenever the corresponding primary tasks are completed successfully. Tercos reduces the scheduling lengths in the fault-free scenario to improve schedulability under executing portions of backup tasks. The second technique (called Debus) schedules backup tasks as late as possible while terminating backup tasks when their corresponding primary tasks are successfully completed, which will further minimize the schedule lengths to enhance schedulability performance. Indeed, Tercos is a passive way of eliminating redundancies of backup tasks, while Debus is a proactive method that defers the execution of backup tasks. The proposed schemes improve the performance of the distributed real-time embedded system while considering the reliability requirement. However, the proposed schemes do not discuss the impact of the overlap minimization through the proposed schemes in power/energy consumption and thermal violation. Moreover, the simulation results are based on random task set generation executing on the Pentium 4 platform.

The proposed method in [43] is a low overhead, semi-partitioned, and optimal fair scheduling technique for the cold standby-sparing (CSS) technique. After detecting a permanent fault, the system boots up the spare core for operation. Hence, it can achieve significantly better resource usage and power efficiencies in comparison with hot standby-sparing. However, similar to any cold standby-sparing based scheme, it must also deal with a recovery period subsequent to a fault, when one less core resource is available. The proposed method distributes the slack times of jobs and minimizes the job terminations and rejections in the recovery phase to maximize the performance of the system. The FT-FS proposed method in [43] evaluates the performance of the CSS proposed technique and its timing overhead in periodic real-time task sets. However, it does not consider the power/energy constraint for safety-critical applications. Moreover, the evaluations are based on simulation.

B. Replication

The authors in [44] have proposed a framework for taking into account the replicated hard real-time tasks in scheduling algorithms that are largely independent of the replication technique (e.g., active, passive, and semi-active replication). They have presented that the way fault-tolerance tasks are integrated into scheduling algorithms depends neither on the type of replication strategy nor on its parameters such as replication level, replication granularity, and replicas' location. They have shown that different replication techniques can be exploited and integrated into a unified scheduling algorithm. The study in [45] has considered only two replicas for each hard real-time task and solved the problem of mapping tasks to a heterogeneous platform in a way that guarantees timing requirements while core failures can be tolerated. In order to solve the task partitioning problem with timing and replication constraints, it has developed a fully polynomial-time approximation scheme. Chen et al. [46] have replicated each hard real-time task on different K processors, where K is a user-defined integer for improving system reliability. Tasks on each processor are scheduled with EDF scheduling. They have presented an approximation algorithm, with a 2-approximation ratio, to minimize the maximum utilization in a system with a

certain number of processors. The proposed approach is then extended to a polynomial-time approximation scheme. Moreover, in order to minimize the required number of processors for feasible scheduling, they have proposed an asymptotic polynomial-time approximation scheme. The proposed methods in [45] and [46] did not contain any practical comparison with other approaches. The work in [47] has considered the problem of maximizing the number of tasks which are successfully assigned to a homogeneous distributed multiprocessor system where the replicas corresponding to the same task are assigned to different processors, and all assigned tasks meet their timing constraints. They have proposed the greedy and polynomial-time approximation algorithms for solving their problem. The effectiveness of this proposed method is evaluated using synthetic generated applications, on a machine (Acer Extensa 5620) that consists of Core 2 Duo 1.83 GHz CPU and 3GB main memory, where fix number of fault occurrence (k=3) should be tolerated. It should be noted that in [45][46][47] the overhead of communication is not considered.

C. Checkpointing (CP)

In the checkpointing technique, the last correct state of the faulty process is stored in the stable memory. Whenever a fault occurs, the system rolls back to the last correct checkpoint and recovers the faulty portion of the task in the last checkpoint duration [16][17]. Although checkpointing increases the execution time of the task in the fault-free scenario, it reduces recovery time when faults occur, since it is not required to re-execute the whole task from the beginning and only the faulty portion is required to be recovered. Note that in real-time systems the location of the checkpoints (intervals between checkpoints) or the size of recovery blocks is of great importance. If the size of recovery blocks is large, it will lead to bigger latency to detect the fault and recover it. On the other hand, if the size of the recovery block is small it will lead to higher overheads in fault-free scenarios. In this regard, the following research [48]-[50] are the example of exploiting checkpointing techniques in real-time embedded systems.

A schedulability test for periodic/sporadic task sets has been proposed in [48], where the task set can be scheduled based on any fixed-priority preemptive scheduling under checkpointing fault-tolerance technique for single-core systems to tolerate transient faults. They claim that the results are applicable to distributed/multiprocessor systems where tasks are statically allocated to individual processors. The proposed checkpointing scheme is an optimal scheduling algorithm for deterministic hard real-time systems. It has implemented an optimization approach based on Tabu search that determines the processes assignment to the heterogeneous nodes, and the assignment of fault-tolerance techniques to processes. The work in [49] has presented a non-uniform checkpointing scheme for soft real-time applications, which is based on a static non-uniform checkpoint placement that asymmetrically stores the processor states. This work supports the non-zero error detection latency. It tries to reduce error recovery latency and the number of checkpoints in order to increase the probability of timely task completion. The proposed method in micro architecture-level is implemented on the VHDL model of LEON2 32 bit processor, which is extended by adding an extra unit, called RUC (Recovery Unit Controller). RUC has been connected to the execution unit, registers file, and data cache memory. To carry out the experiments to evaluate the effectiveness of the proposed method, four benchmarks including bitcount, basicmath, bubble sort, and matrix multiply from MiBench suit are selected. Zhengyong et al. [50] have provided a scheduling analysis to deal with the burst fault model and determine the optimal number of checkpoints for hard real-time tasks to minimize the worst-case execution time of tasks in presence of faults. The tasks can be scheduled based on a fixed-priority algorithm such as rate monotonic or deadline monotonic scheduling algorithm, which assigns different priorities to each task. The burst fault model defines processes that can cause random faults over a short period of time. Moreover, it has exploited task reallocation to deal with permanent faults. The applicability of the proposed approach is evaluated by generating synthetic task sets.

It is worthy to mention that the aforementioned proposed methods did not consider the effect of checkpointing on the power/energy consumption of the system.

D. N Version Programming (NVP)

A greedy task mapping, called dTune (dependability Tuning), has been proposed in [51] to improve the reliability of the multiprocessors considering, aging, process variation, and soft errors. This method efficiently selects suitable code versions using the knowledge of on-chip process variation and performance variations due to aging effects at runtime. The authors consider reliability-driven compiler results to determine the reliability of the different code versions of the applications. Moreover, they have developed a processor aging estimator, and considered the effects of process variations and aging. Since in a real-world scenario, an on-chip many-core system is susceptible to multiple of such reliability threats and different cores may experience different performance variations and soft error rates, joint consideration of aging and process variations is important. In this paper, a many-core processor with N ISA-compatible homogenous RISC cores (e.g., a 5-stage pipeline LEON3 embedded processor) is considered. Note that due to the design-time process variations or run-time NBTI aging effects, heterogeneous w.r.t. their performance capabilities will be considered at runtime. The work in [52] has presented an enhanced redundancy technique for multi-core systems executing software replicas. It tries to recover permanent faults by introducing a new cost-efficient software diversity technique. For example, different compilers can be used to generate binaries for the same program that means different characteristics in the execution of the same calculations. The presented solution in this paper claims that its method independent of the type of processors.

E. Re-execution

The study in [53] guarantees the timing constraints in the fault occurrences scenario for hard real-time tasks while maximizing the overall utility of soft real-time tasks. This method has two offline and online scheduling. Offline scheduling is not fault-tolerance and is pessimistic from the utility perspective. However, the online approach calculates a new schedule each time a process encounters a fault or completes without producing acceptable output. The proposed method synthesizes a set of schedules at the offline phase which is known as a quasi-static scheduling strategy. Then, in the online phase, based on the fault occurrence scenario, the correct schedule will be selected. Synthetic applications with a random number of processes are generated to evaluate the proposed method. It has been considered that the proposed method can tolerate three fault occurrences. The experiments have been run on a Pentium 4 with 2.8 GHz processor with 1Gb of memory. The authors in [54] have proposed models to evaluate the effect of fault detection and recovery strategy on timing constraints by considering different fault models (such as single or multiple fault occurrence), task execution model, and hardware platform (such as several types of single-core, multi-core, and distributed platforms). In order to analyze the feasibility, it has exploited mixed-integer linear programming (MILP) for joint task allocation and scheduling with tolerance mechanisms under timing and fault tolerance constraints. Furthermore, it has presented a Monte Carlo based simulator to check fault coverage of the system and meeting timing constraints. The proposed approach is applied to the synthetic task sets (which are generated with TGFF tool) and industrial case study (which is derived from a subsystem of an experimental vehicle).

The exploitation of re-execution fault-tolerance technique in mixed-criticality systems is presented in [55]-[59]. The proposed method in [55] is a mixed-criticality fault-tolerance scheme that maximizes the utilization of the system while preserving the reliability at the guaranteed level. It exploits an optimization theory to find the proper slack windows for the execution of both critical and non-critical tasks, which maximizes the utilization of each node while preventing overloads. For this purpose, the tasks are distributed to the nodes regarding minimizing the overhead of nodes, and the feasible scheduling avoids any interference between the execution of the critical and non-critical tasks. Moreover, in each node, the Integer Linear Programming (ILP) aims for tasks to be executed based on their priority while reducing the execution cost, and guarantees the feasibility and reliability of tasks. In order to control the effect of a node failure, the alternate versions of certain critical tasks are scheduled on different cores. The study in [56] has proposed an exact feasibility test which is necessary and sufficient for a set of periodic tasks to tolerate faults on a single-core processor. The proposed feasibility test is applicable to any fixed-priority scheduling algorithm such as rate-monotonic or deadline-monotonic algorithms. They have considered multiple faults can occur on each task at any time, even during the recovery process. The proposed feasibility test can tolerate the maximum number of faults that can occur at any time interval. The work in [57] has proposed a new model for mixed-criticality systems from the perspective of fault tolerance, in a single-core processor. It has first scheduled the primary task, and if the primary task encounters a fault, the backups are dispatched one by one until the correct output is reached. It has also derived a sufficient schedulability test for a fixed-priority scheduling algorithm that guarantees meeting all deadlines even if backups are executed to recover from faults. The efficiency of the proposed method is examined through synthetically generated task sets. The study in [58] addresses overrun and fault occurrence with separate operational modes in a single-core and multi-core processor while executing as many low-criticality tasks as possible in each system operational mode. The experiments are conducted based on synthetically generated tasks' set. An approach to increase the lifetime of mixed-criticality systems while satisfying the timing and safety requirements has been presented in [59]. This work has considered transient and permanent faults that are caused by thermal cycling. Moreover, it has presented two MILP-based methods to solve the scheduling problem and a time-efficient CEM-based (cross-entropy method) heuristic for maximizing the lifetime of the system. Simulations are carried out based on the synthetic generated benchmarks, and real-world benchmarks. A random task generator is used to produce multiple synthetic benchmarks (i.e., task sets). Moreover, five real-world benchmarks from flight management systems are tested. The Alpha 21264 processor (with complete power and thermal models) is exploited as the hardware platform.

F. Combination of Several Fault-Tolerance Techniques

The work in [60] has considered the problem of analysis and optimization of fault-tolerance hard real-time task scheduling for multiprocessor embedded systems. In order to compute the system-level reliability, a Binary Tree Analysis (BTA) in the presence of re-execution and hardware replication techniques is proposed. By integrating the analysis with optimization based on the Multi-Objective Evolutionary Algorithm (MOEA), the feasible schedules under reliability, resource, and timing constraints are synthesized. Tasks' mapping to processors, the assignment of the fault-tolerance policy, and tasks' scheduling are the output of the optimization algorithm. They have considered an application as the functionality of the system that consists of a set of independent jobs, each given as a directed acyclic graph. In this paper, multi-processor systems are considered. In multi-processor systems, if two dependent tasks are mapped to different processors, a message must be scheduled for data transfer between processors. Hence, the latency of message transfer is also considered. The study in [61] has developed analysis and optimization techniques that consider imperfect fault detection and distinguishes detectable and undetectable faults in the overall workflow. It exploits both temporal and

spatial redundancies in hard real-time systems. In addition, it has proposed an approach based on the multi-objective evolutionary algorithm (MOEA) for reliability-aware design optimization. The target architectures of [60] and [61] are heterogeneous multi-processor platforms consists of two types of Processing Elements (PEs), namely a RISC processor and a DSP with time-triggered communication. The communication between tasks is implemented with messages. Pop et al. [62] have considered hard real-time safety-critical systems which are scheduled based on static cyclic scheduling. The study in [63] has generalized the proposed model in [58] for mixed-criticality systems to support on-demand redundancy which exploits dual modular redundancy (DMR), TMR, and passive replication. Indeed, the proposed method improves the QoS of low-criticality tasks. The effectiveness of the proposed idea is evaluated using synthetic generated applications.

IV. Power/Energy-Aware Fault-Tolerance Techniques

In this section, proposed mapping/scheduling techniques are categorized based on their employed fault-tolerance technique. Table 2 shows an overview of the proposed approaches which consider power/energy in designing fault-tolerance real-time embedded systems.

A. Standby-Sparing (SS) and Primary/Backup (P/B)

An application-level fault-tolerance (ALFT) approach has been proposed by Unsal et.al [64] to evaluate the energy efficiency and fault-tolerance technique simultaneously in hard real-time systems. The ALFT employs a primary and secondary approach (to tolerate one permanent fault) and tries to complete the execution of the primary task as soon as possible, and delay the execution of the secondary task to diminish the execution overlap between the primary and secondary tasks. In this regard, the ALFT heuristic uses an energy-efficient Earliest Deadline First (EDF) and Shortest Execution-time First (SEF) scheduling algorithms and evaluates the efficiency of each approach to achieve an energy-efficient fault-tolerance technique. Although the proposed method attempts to reduce energy consumption by eliminating the overlap between the execution of the primary and backup tasks, however, it does not employ a power/energy management technique (e.g., DVFS). Moreover, due to not considering the peak power constraint, it may face temperature violation. It should be noted that the proposed method is a software-level approach, which does not need any hardware modification. The effectiveness of the proposed method is evaluated through generating random tasks. Although, it does not consider any real-world platform for evaluations. In order to minimize the energy of standby-sparing, Ejlali et al. [65] have applied DPM on the backup core, and dynamic voltage scaling (DVS) on the primary core. This study has shown the negative impact of the DVS technique on reliability. Hence, it proposed an analytical approach to assign the proper supply voltage value of the primary core at runtime to reduce energy consumption by exploiting dynamic slack times and meets the reliability target. The presented method in [66] is an online energy management technique for a standby-sparing system that considers the overheads of activation and voltage transition forced by DPM and DVS techniques, respectively. Moreover, the energy manager is considered as a task and its overhead is computed. Indeed, this method exploits dynamic released slack times at runtime to reduce energy consumption while guaranteeing timing constraints. The proposed methods in [65] and [63] exploit RTEMS real-time operating system in the ARM7TDMI-based system and select the tasks from the MiBench benchmark suite for evaluation.

The study in [67] has considered the effect of frequency scaling on the fault arrival rates. Therefore, they have proposed the enhanced primary/backup model, where accounts for the loss of reliability due to frequency scaling by scheduling additional copies of tasks. In this approach, the switch to a lower frequency is committed only if the available slack is large enough to accommodate the additional copies required to retain the original reliability provided before frequency scaling. Due to the iterative selection of proper frequency for the execution of the tasks, time overhead and probability of failure will be increased. Moreover, reducing the energy/power consumption without consideration of peak power constraints faces thermal violation. The proposed method considers the Intel Xscale and Transmeta Crusoe processors for the evaluation of the proposed method in a homogeneous and heterogeneous system. Nevertheless, the task sets are generated synthetically, and execution times are assumed to have a normal probability distribution function.

The study in [68] has proposed two energy-aware fault-tolerance scheduling algorithms based on a primary-backup approach called “Fault-tolerant Energy Efficient task scheduling with Delayed and Overloaded backups (FEED-O)”, and “FEEDO with Dynamic-deferring (FEED-OD)”. Primary tasks will be executed on a DVS-enabled processor, while the backup copies are scheduled on the auxiliary processor with DPM for reducing energy consumption. It has proposed the overloading of backup jobs and analyzed it for reducing the energy consumption of scheduling periodic task-sets with rate monotonic scheduling. Indeed, in order to ensure that all backup jobs will not be executed at runtime on an auxiliary processor, they are scheduled in an overlapped time interval, which is called backup-overloading. If overloading of backup jobs on an auxiliary processor is done, then it may further enhance energy saving by deferring the start time of backup copies.

Paired-Standby-Sparing (Paired-SS) and Generalized-Standby-Sparing (Generalized-SS) have been proposed in [69] and [70]. In the Paired-SS method, cores are divided into pairs, and the standby-sparing method is applied to each pair. On the other hand, the Generalized-SS divides the cores into two categories, primary and secondary cores. The primary tasks are executed in the primary cores under partitioned-EDF and the DVFS technique. The backup

tasks are executed in secondary cores under partitioned-EDL with the DPM mechanism. The 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. A low-energy task scheduling algorithm that employs the adaptive dual-queue mechanism to postpone backup tasks' execution has been introduced in [71]. The primary tasks are scheduled by the EDF algorithm to minimize execution overlap between primary and backup tasks. The studies in [66], [70], and [71] employ a discrete event simulator. Therefore, there is not any consideration for the platform model, and the evaluations ignore some system overheads.

The standby-sparing technique is exploited by Roy et al. [72] to tolerate both permanent and transient faults for heterogeneous multi-core systems, where the platform includes high-performance (HP) and low-power (LP) cores. The proposed method in [72] determines the proper type of cores for primary and backup tasks to minimize energy consumption. To further reduce energy consumption, the primary core uses DVFS (the proper frequency level for the primary core is determined), while the spare one employs DPM. The proposed method considers an ARM big.LITTLE heterogeneous platform. However, in order to evaluate the effectiveness of the proposed method task sets are generated randomly. Therefore, the impact of system configuration on the tasks' specifications is not considered.

A shared resource standby-sparing scheme to preserve the original reliability for the dynamic-priority real-time task has been presented in [73], which schedules tasks according to the Earliest Deadline First/Dynamic Deadline Modification (EDF/DDM) policy. Moreover, primary and backup tasks can be executed at a uniform speed. In addition, it exploits the mixed mapping partitioning method, in which the tasks need to access the shared resources are assigned to the primary cores, and other tasks are assigned to the spare cores. Moreover, to save energy, DVS and DPM techniques are applied to both primary and backup tasks.

Ansari et al. [74] have proposed a method that uses the standby-sparing technique for periodic real-time tasks to satisfy the Thermal Design Power (TDP) constraint. In this method, the primary and backup cores exploit peak-power-aware EDF, and peak-power-aware EDL policies, respectively. In order to reduce the peak power consumption, the proposed method delays the execution of backup tasks as much as possible and tries to cancel the execution overlap of backup tasks. It exploits gem5 and McPAT simulators to evaluate the proposed method. Moreover, tasks are selected from MiBench benchmark suite in an ARM-based system.

The work in [75] has proposed two schemes (known as MC-2S and MC-4S) to tolerate permanent faults through applying the standby-sparing technique with low energy overhead in mixed-criticality systems. In both schemes, two copies of each high-criticality task are scheduled on different cores to guarantee their timeliness in case of permanent fault occurrence. In order to guarantee the quality of service of low-criticality tasks, in case of permanent fault or overrun occurrence two different strategies are proposed. In the MC-2S scheme, sufficient slack time is reserved to schedule a backup task for each low criticality task on an alternative core. However, the MC-4S scheme exploits semi-partitioned scheduling in which the low-criticality tasks migrate to other cores. The schedulability analysis to guarantee the timeliness, and QoS in the proposed algorithm along with the reliability-aware DVFS method is approved through demand bound function analysis. Due to lack of benchmark for mixed-criticality systems, the studies in [72] exploit synthetic generated task sets for evaluation. The study in [76] has introduced a parallelism and reduction policy in every primary-backup pair of the multi-core platform to increase the quality of service (QoS) of low-criticality tasks in a mixed-criticality system. It also minimizes the energy consumption through convex optimization, and proposes a heuristic for energy reduction by reducing the execution time and overlap between primary and backup tasks.

Ansari et al. [77] have proposed a peak-power-aware primary/backup scheme for frame-based soft real-time tasks. The proposed scheme removes the peak power overlaps of concurrently executing tasks to reduce the peak power consumption and meet the chip-level TDP constraint. To do this, the proposed method receives the tasks' power profiles, and presents a task partitioning method, and two developed scheduling policies known as maximum-peak-power-first and maximum-peak-power-last to schedule primary and backup tasks, respectively. This method receives the power and performance information of LEON3 processor from a logic simulation. After that, a software-level simulation and fault injection are used to evaluate the proposed method with the MiBench benchmark task sets.

B. Replication

Assayad et al. [78] have proposed a heuristic-based scheduling algorithm considering three criteria: a) minimizing the length of the schedule in real-time systems, b) maximizing the reliability in dependable systems, and c) minimizing energy in autonomous systems. Their proposed method maximizes reliability through active replication and employs DVS to reduce power consumption. In this regard, the proposed scheduling algorithm divides the problem of simultaneous reliability improvement, power reduction, and minimum scheduling length into the reliability improvement and power minimization problem for each defined cell of the grid, and after that, the scheduling part reduces the scheduling length of the cell. The goal of [79] is to find the proper number of replicas, frequency assignment, and core allocation for each periodic hard-real time task at the offline phase to achieve the reliability target while minimizing the overall energy consumption of the system. At run-time, they find the first copy of tasks that have been completed successfully and cancel the execution

of their other replicas to achieve even more energy saving. Note that the proposed system supports core-level DVFS to further reduce energy consumption.

Spasic et al. [80] have proposed a polynomial-time solution approach which efficiently maps and schedule hard real-time streaming applications onto clustered heterogeneous MPSoCs such that the required throughput is satisfied and the energy consumption is minimized through per-cluster voltage frequency scaling (VFS). Moreover, it determines the required number of replicas for each task in a graph, and balance the distribution of the tasks on the same type of processors, to reduce the energy consumption through running processors at lower voltage and frequency levels. Tasks are scheduled based on EDF policy. The experiments were performed on the real-life applications from the StreamIt benchmarks suit.

The study in [81] shows how task replication can be exploited to satisfy a given task-level reliability target that is expressed in terms of tolerating transient faults. Moreover, the fault coverage factor of the fault detection techniques is taken into account. By considering the negative effect of DVFS on the rate of transient fault, it has presented a technique to compute the number of replicas and the frequency assignment for each task while minimizing the overall energy of the hard real-time system. Moreover, in order to reduce the execution overlap between the primary task and its corresponding replica(s) it has developed a static solution and dynamic adaptation. In order to evaluate the effectiveness of the proposed method synthetic tasks set based on UUnifast algorithm are generated and executed on 4- to 12-core processors.

The authors in [82] aim to propose energy-efficient scheduling by considering a reliability target (known as ESRG algorithm) to minimize the energy consumption while meeting the reliability target for the parallel applications. Moreover, they further proposed energy-efficient fault-tolerance scheduling with a reliability goal method (known as EFSRG) to minimize the energy consumption while meeting the reliability target based on an active replication scheme when the application's reliability target is unreachable. Both proposed methods are solved through three steps: *i*) Prioritizing tasks: Prioritizing tasks problem is an important problem for DAG list scheduling on heterogeneous distributed systems. Among existing prioritizing task schemes, this paper considers the descending order of upward rank value of tasks as the criterion for DAG list scheduling, because it has been widely used in energy-efficient and reliability-aware scheduling. *ii*) Satisfying reliability goals: The reliability value of an application is the product of the reliability value of each task. Therefore, if the reliability of all tasks exceeds the reliability target, then the reliability value of the application must exceed its reliability goal. *iii*) Reducing energy consumption: Each task only selects the processor and frequency combination with the minimum dynamic energy consumption while satisfying its reliability target.

The LETR-MC scheme has been proposed in [83] that satisfies timing, energy, reliability, and service level constraints in mixed-criticality multi-core systems. The task replication is employed to satisfy reliability requirements and enhance the QoS of low-criticality tasks in the overrun operational mode of the system. The proposed scheme computes the minimum number of replicas for each high-criticality task to meet the reliability target. It has developed a unified demand bound function analysis to check the schedulability and applying DVFS for energy reduction. Tasks can be assigned based on worst-fit, best-fit, or first-fit bin packing strategy, and they are scheduled based on proposed ER-POED (ER-Preference-Oriented Earliest-Deadline first) scheduling algorithm on multi-cores.

Saber-Latibari et al. [84] have proposed a mapping and scheduling method by employing task replication mechanism for a task-graph model of applications in heterogeneous multi-core systems. The hardware configuration of this work is a processor with two heterogeneous islands which execute a different number of tasks. The cores on the high-performance island are considered to be Alpha21264 type, with lower execution time of tasks and higher reliability, while the cores on low power island are considered ARM Cortex-A15 with higher execution time of tasks and less power consumption.

Yeganeh-Khaksar et al. [85] have presented a novel mapping and scheduling method for the problem of achieving the desired reliability target that meets the chip-level power constraint. In the proposed scheme, first, tasks are assigned based on reliability-aware lowest utilization policy, then, tasks are scheduled based on maximum-power-aware EDF policy, and finally, the reliability-and-peak-power-aware DVFS technique is employed for meeting TDP constraints. The effectiveness of the proposed method is evaluated based on an ARM processor with core-level DVFS capability and there are 6 different frequency/voltage levels from [0.85Volt, 1GHz] to [1.1Volt, 2GHz] to reduce peak power consumption.

As it can be seen in the mentioned related work, the works that consider the replication technique reduce power and energy consumption through applying DPM and DVFS in different types of platforms, but they did not discuss temperature issues. It is worthy to mention that inserting more replicas will increase the extra power consumption on the chip, and thereby on-chip temperatures might increase beyond safe limits. Therefore, in order to prevent thermal violations, it is indispensable to consider a temperature constraint for the task replications techniques.

C. N Modular Redundancy (NMR)

An Optimistic Triple Modular Redundancy (OTMR) scheme has been introduced in [86] which reduces the energy consumption of conventional TMR systems. In OTMR one of the processing units is turned off or slowed down, however, if the other two units encounter a fault, it should boot up and complete the computation before the timing constraint. The optimal frequency assignment for OTMR was discussed in [87]. The study in [87] has been

explored the optimal frequency setting to minimize the system energy consumption of the OTMR scheme and has been considered a single task within each frame. Indeed, it analytically compared the OTMR, conventional TMR, and classical DWC. It is worthy to mention that the studies in [86] and [87] don't need any additional hardware modification, however, they need pre-defined synchronization points. Moreover, for the evaluation, they generate synthetic tasks set with an Intel Pentium3 processor.

The work in [88] has proposed an aging-aware adaptive fault-tolerance method called ANMR for DVFS-enabled

Table 2. Summary of power/energy-aware fault-tolerance techniques in real-time embedded systems.

Ref.	Application Model	Architecture Model	Fault-Tolerant Technique	Goals/Constraints	Energy Management Technique
[64]	Periodic	Homogeneous Multi-core	Standby-Sparing (SS), Primary/Backup (P/B)	Energy/Timing/Reliability	-
[65][66]	Frame-based	Homogeneous Dual-core		Energy/Timing/Reliability	DVS, DPM
[67]	Periodic	Homo. & Hetero. Dual-core		Energy/Timing/Reliability	DVS
[68][69]	Periodic	Homogeneous Multi-core		Energy/Timing/Reliability	DVS, DPM
[70]	Periodic	Homogeneous Multi-core		Energy/Timing/Reliability	DVFS, DPM
[71]	Periodic	Homogeneous Dual-core		Energy/Timing/Reliability	DVS, DPM
[72]	Frame-based	Heterogeneous Dual-core		Energy/Timing/Reliability	DVFS, DPM
[73]	Periodic	Homogeneous Dual-core		Energy/Timing/Reliability	DVS, DPM
[74]	Periodic	Homogeneous Multi-core		Energy/Timing/Reliability/Peak power	DVFS, DPM
[75]	Periodic	Homogeneous Multi-core		Energy/Timing/Reliability/QoS	DVFS
[76]	DAG	Homogeneous Multi-core		Energy/Timing/Reliability/QoS	DVFS, DPM
[77]	Frame-based	Homogeneous Multi-core		Timing/ Reliability/Peak power	DPM
[78]	DAG	Homogeneous Multi-core	Replication	Energy/Timing/Reliability	DVS
[79]	Periodic	Homogeneous Multi-core		Energy/Timing/Reliability	DVFS, DPM
[80]	DAG	Homo. & Hetero. MPSoC		Energy/Timing/Reliability/Throughput	VFS
[81]	Periodic	Homogeneous Multi-core		Energy/Timing/Reliability	DVS
[82]	DAG	Heterogeneous Multi-core		Energy/Timing/Reliability	DVFS
[83]	Periodic	Heterogeneous Multi-core		Energy/Timing/Reliability/QoS	DVFS, DPM
[84]	DAG	Heterogeneous Multi-core		Reliability/QoS/Power	DFS, DPM
[85]	Periodic	Homogenous Multi-core		Energy/Timing/Reliability/TDP	DVFS, DPM
[86]	Periodic	Single-core	N Modular Redundancy (NMR)	Energy/Timing/Reliability	DVS, DPM
[87]	Frame-based	Single-core		Energy/ Timing/Reliability	DVS, DPM
[88]	Frame-based	Homogeneous Multi-core		Energy/ Timing/Reliability	DVFS
[89]	Frame-based	Homogeneous Multi-core		Energy/ Timing/Reliability	-
[90]	Frame-based	Homogeneous Multi-core		Energy/ Timing/Reliability	DVFS, DPM
[92]	DAG	Homogeneous Multi-core		Energy/Timing/Reliability/TDP	DVFS, DPM
[91]	DAG	Homogeneous Multi-core		Energy/ Timing/Reliability	DVFS
[93]	DAG	Homogeneous Multi-core		Power/ Timing/Reliability	DVFS
[94]	Periodic	Single-core	Checkpointing (CP)	Energy/Timing/Reliability	DVS
[95]	DAG	Homogeneous Multi-core		Energy/ Timing/Reliability	DVS
[96]	Aperiodic	Single-core		Energy/Timing/ Reliability	DVS
[97]	Frame-based	Homogeneous Multi-core		Energy/Timing/Reliability	DVFS
[98]	Periodic	Single-core		Energy/Timing/Reliability	DVS
[99]	Periodic	Homogenous Multi-core		Energy/Timing/Reliability	DVS
[100]	Aperiodic	Single-core		Energy/Timing/Reliability	DVFS
[101][102]	Frame-based	Homogenous Multi-core		Energy/Performance/Timing/Reliability	DVFS
[103]	Frame-based	Single-core		Energy/Timing/Reliability	-
[104]-[106]	Frame-based	Single-core	Re-execution, Shared Recovery	Energy/Timing/Reliability	DVFS
[107][108]	Periodic	Single-core		Energy/Timing/Reliability	DVFS
[109]	Frame-based	Heterogeneous Multi-core		Energy/Timing/Reliability	DVFS
[110][111]	DAG	Heterogeneous Multi-core		Energy/Timing/Reliability	DVS
[112]	Multiple DAG	Homogeneous Multi-core		Energy/Timing/Reliability	DVFS
[113]	Frame-based	Homogeneous Multi-core		Energy/Timing/Reliability	DVFS
[114]	DAG	Heterogenous Multi-core	Replication, Re-execution	Energy/Timing/Reliability	-
[115]	DAG	Homogeneous Multi-core	P/B, CP	Energy/Timing/Reliability	DVFS

multi-core embedded systems. The ANMR assumes that applications in a system or functions of the same application may have different vulnerability levels, and consequently, their reliability levels are different. ANMR determines the generated slack time when the current frequencies of some cores change from scaled value to maximum one. It checks whether generated slacks are enough to execute the redundant copies of the critical task in parallel with the main one. The selection of cores for mapping tasks is combined with the DVFS technique such that the energy consumption of the system is kept near its original value. In this way, it needs additional hardware to keep the history table. The effectiveness of the proposed method is evaluated based on real-world applications from the ParMiBench suite with LEON3 processor. The study in [89] has presented a low-energy NMR based on approximate computing. Redundant tasks are executed based on approximately reduced execution time and energy consumption. Therefore, in addition to increasing the reliability of a system, instead of exploiting N cores in traditional NMR, only k cores are deployed, where $k<N$. However, it needs special software for comparison. The effectiveness of the proposed method is evaluated based on AxBench and MiBench benchmark suites running on ARM7-based homogenous processors. The energy-budget-aware reliability management (enBudRM) scheme with hybrid energy sources (consisting of renewable and non-renewable energy sources) has been studied in [90]. In the offline phase, the battery is the only energy source of the system and, based on the available budget of energy and static slack time, voltage-frequency levels are determined and tasks are scheduled. In the online phase, to increase the system lifetime, released dynamic slack times are used to further energy saving. To compensate for the reliability loss of DVFS, an energy harvester is considered along with the battery to enable executing more task replicas.

The technique has been proposed in [91] to execute the hard real-time tasks based on the NMR technique in the indispensable and on-demand phases. In this work, the system starts execution in the indispensable phase. If a task has no faults during the indispensable phase, the time which is reserved in advance for its corresponding copies to execute in the on-demand phase is released to employ for significantly reducing power and energy. It exploits MiBench benchmark suite with Intel PXA270 processor to evaluate the effectiveness of the proposed method. Ansari et al. [92] have developed a scheduling algorithm based on the peak-power-aware longest task first policy to prevent overlaps of the concurrently executing tasks to meet the TDP constraint. To reduce the instantaneous power dissipation the DVFS technique is applied on each core. To further reduce the power in the realistic scenario the task cancelation scheme is employed. The proposed method is evaluated based on MiBench benchmark suite with ARM7 processor. The study in [93] has presented a power-efficient reliability management technique which exploits DMR and TMR techniques operating in different voltage and frequency levels. It also considers diversities in execution time properties, software vulnerability, and process variations in hardware.

Finally, it should be noted that all the mentioned approaches did not consider thermal or peak-power constraints. Therefore, in scenarios that all cores will be activated simultaneously to execute tasks, the temperature constraints will be violated.

D. Checkpointing (CP)

Zhang et al. have studied checkpointing in [94], and [95]. The authors in [94] have presented a feasibility analysis for checkpointing schemes in fixed-priority hard real-time systems with constant and variable processor speeds. DVS is then applied based on the feasibility analysis. Important practical challenges such as fault occurrence during checkpointing, the overheads of accessing memory, rollback recovery, the energy of checkpointing, the energy of DVS, and context-switching are discussed in this paper. The Intel XScale PXA260, and the Transmeta Crusoe which are low-power embedded processors are selected for the experiments. The proposed schemes are evaluated based on three real-life task sets, including a CNC, an inertial navigation system (INS), and a generic aviation platform (GAP) task sets. Since the number of tasks for CNC and INS is relatively small, the simulation results for CNC and INS are obtained using the exhaustive search method. The simulation results for GAP are obtained using the heuristic method. The task execution times for these task sets are assumed for a nominal CPU frequency of 200 MHz.

The study in [95] has developed schedulability tests for the checkpointing scheme in a DAG task model for a constant speed and DVS-enabled processor. It deterministically guarantees the timeliness of tasks even when transient faults occur. It considers that the system is composed of processing elements and all of them take checkpoints simultaneously under a global clock in a message-passing system. It also considers that at most k faults can occur in the system before the overall deadline of the graph. The goal of this paper is to find the distance between checkpoints and proper voltage assignment to each task for saving energy. The effectiveness of the proposed method is evaluated by executing E3S benchmark set on the AMD K6 processor. The benchmarks are based on the Embedded Microprocessor Benchmark Consortium (EEMBC) and include tasks in the application domains of automotive systems, telecommunications, and consumer electronics.

The study in [96] has proposed two policies for checkpoint placement in aperiodic tasks and computed the optimal number of checkpoints from the energy consumption perspective in each proposed method. It exploits slacks in the schedule for applying the DVS technique to further reduce energy consumption while tolerating faults. It is assumed that one fault can occur during the execution of each task and the processor can scale its frequency in a continuous range. Simulation experiments are conducted using the Simple-Scalar

simulator by executing Li, Perl, Go, and Compress programs from the SPEC benchmarks. Zhu et al. [97] have investigated the effect of DVFS on the reliability of the system. Their evaluations have shown that the reliability of the system depends on the number of recoveries before the deadline of the applications. In order to analyze the effect of frequency and voltage scaling on fault rate, simulation-based on Intel Pentium M processor with RAMBUS memory are conducted.

Wei et al. [98][99] have studied energy management in the checkpointing technique. The quasi-static task scheduling for fixed-priority rate monotonic algorithm (RMA) has been proposed in [98], which is composed of offline and online phases. This paper aims at reducing energy by designing an efficient offline algorithm that can be adapted based on the runtime behavior of the system. It has proposed two offline energy management techniques known as application-level DVS where all the tasks share the same voltage level; and task-level DVS where each task is executed based on its voltage level. Moreover, the feasibility test of RMA algorithm is achieved through an exact timing analysis approach. The energy efficiency of the proposed scheduling algorithms in [98] is evaluated through simulation based on Transmeta Crusoe with 5 voltage and frequency levels, and Intel XScale PXA260 with 3 voltage and frequency levels processors. Moreover, two real-life task sets including Inertial Navigation System (INS), and Computer Numerical Control (CNC), are exploited for evaluation. The online overhead of the proposed schemes was also evaluated using a SimpleScalar-based Intel XScale processor simulator. The study in [99] has proposed energy-efficient task allocation and scheduling schemes to minimize energy through the DVS technique and maximize performance in checkpointing strategy on hard real-time homogenous symmetric multiprocessor (SMP) systems. The proposed heuristic achieves energy saving by optimally balancing the application workload among processors in a system where processors have identical characteristics, and support continuous voltage scaling. The simulations were repeated for varying numbers of processors between two processors to eight processors. Tasks are generated randomly.

The proposed method in [100] has considered the reliability of the system, application size, and fault rate in the proposed scheme to reduce energy consumption dynamically. In this scheme, the slack times are used to save energy without decreasing the reliability by scheduling an additional recovery task. Furthermore, the checkpointing technique is employed to use slack times more efficiently, especially in the case where the slack time is not enough for the recovery of a whole task. The evaluations are conducted based on randomly generated task sets.

Han et al. have exploited the checkpointing technique in [101] and [102]. The study in [101] at first proposed an optimal checkpointing scheme that minimizes the worst-case response time of tasks set on a single-core processor; that all tasks share the same reserved recovery. Then it allocates tasks to multiprocessors in a way the energy consumption is reduced. In this step, it shows that resource reservations on each processor for optimal checkpointing and exploiting slack time for applying DVFS are in contrast with each other, and solves this problem. The effectiveness of the proposed method is evaluated with randomly generated task sets. The study in [102] has proposed an energy-efficient method that solves the problem of simultaneous fault-tolerance and task mapping under the DVFS condition for periodic fixed-priority hard real-time tasks. It selects the proper number of checkpoints to check the schedulability of tasks and guarantees the fault tolerance of the system, then maps the tasks under DVFS and checkpointing conditions to minimize energy consumption. At first, the effect of the increasing number of tolerable faults, and the effect of increasing checkpointing overhead on the timing complexity of the proposed method is evaluated on a uniprocessor platform based on randomly generated task sets. Then the effectiveness of the proposed energy-saving algorithm is evaluated through simulations.

A two-state checkpointing (TsCp) scheme has been proposed in [103], which considers two operational modes for the system; i.e., fault-free and faulty. It finds non-uniform (non-equidistant) and uniform (equidistant) checkpoint intervals for fault-free and faulty states, respectively. In the offline phase, the optimized checkpoint intervals are computed for all fault occurrence scenarios. In the online phase according to the fault-occurrence scenario, the checkpoint intervals which are computed in the offline phase, are selected. From the beginning of the execution of a task, non-uniform checkpointing is used. As soon as the first fault occurs, the remaining execution time of the task will be executed based on uniform intervals. In order to minimize the number of checkpoints in fault-free states, checkpoint insertions are postponed as much as possible, however, enough recovery time for the faulty state is considered. In the faulty state, the optimal number of uniform checkpoints is computed based on [48]. Experimental evaluation of this method is conducted based on executing various real-life applications on a system-level simulator. Power and performance characteristics are obtained from LEON3 processor and an emerging NVM technology. The applications and processor characteristics were obtained through detailed ASIC synthesis and gate-level simulations. The memory characteristics were obtained from the NVsim tool.

E. Re-Execution

The study in [104] has developed two polynomial-time heuristic schemes to solve the slack allocation problem for minimizing energy while preserving the reliability target. The proposed method in [105] has exploited the shared recovery technique to minimize the energy while simultaneously satisfy the original reliability of the system. Indeed, in this method instead of allocating separate recovery to each task at the offline phase, a global or shared recovery block can be used by all tasks at the online phase. The presented method in [106] has formulated the

Table 3. Summary of temperature-aware fault-tolerance techniques in real-time embedded systems.

Ref.	Application Model	Architecture Model	Fault-Tolerant Techniques	Goals/Constraints	Energy Management Technique
[116]	Frame-based	Heterogeneous Multi-core	Replication	Timing/Reliability/Energy/Temperature	-
[117]	Periodic	Heterogeneous Multi-core	Replication	Timing/Reliability/Energy/Temperature	DVS, DPM
[118]	Frame-based	Single-core	Checkpointing	Timing/Reliability/Energy /Temperature	DVFS
[119]	DAG	Homogeneous Multi-core	NMR	Timing/Reliability/Power/Temperature/QoS	DVFS
[120]	DAG	Heterogeneous Multi-core	Cold Standby-Sparing	Timing/Reliability/Power/Temperature/QoS	DVFS, DPM
[121]	Periodic	Heterogeneous Multi-core	Replication, NVP	Timing/Reliability/Power/Temperature	DFS, DPM

reliability-aware energy management problem for a set of frame-based task models as a nonlinear optimization problem and obtains the static optimal solution. Then, it has adjusted the task frequencies at runtime by detecting dynamic slacks through early completion of tasks, and proposed an online algorithm to improve the overall reliability. Moreover, the proposed solutions are extended to the periodic task model.

A reliability-aware energy management technique has been presented in [107] for a set of periodic tasks which are scheduled based on EDF policy. In order to solve the problem, it has proposed two task-level heuristics. Then, it introduced a wrapper-task mechanism to monitor dynamic slacks at runtime and manage them efficiently through the job-level online scheme in a reliability-aware manner. Indeed, whenever recovery tasks/jobs are required for preserving reliability, they are scheduled dynamically at the task's dispatch time. Otherwise, the remaining slack is exploited for energy savings. The study in [108] has proposed an approach based on dynamic allocating recoveries to achieve optional reliability levels for each periodic task, when employing DVFS. It has also developed a pseudo-polynomial time feasibility test for the proposed method. Qi et al. [109] have derived a reliability-aware global scheduling scheme to reduce the energy consumption for a frame-based task model. They have considered that different tasks can share the same reserved slack time to recover from faults. Then, in case of fault occurrence, the entire faulty task has to be re-executed.

Pop et al. [110] have presented a scheduling and energy management method for hard real-time tasks mapping on distributed heterogeneous embedded systems. It has proposed a constraint logic programming-based algorithm which can synthesis the fault-tolerance schedules for finding schedulable and reliable results within limited hardware and energy resources. An optimization scheme for joint reliability and energy management in DAG applications through adopting the shared recovery technique has been proposed in [111]. To solve the mentioned problem, this paper has proposed three algorithms that each of them consists of three steps: *i*) Task priority establishment phase, ii) Frequency selection phase, and iii) Processor assignment phase. Finally, the task recovery phase detects the transient fault and recovers the faulty task, to reach higher reliability with lower total energy. The work in [112] has considered a hybrid scheme at both design-time and run-time for reliable and low-energy mapping and scheduling of dependent applications, in multi-core embedded systems with solar energy harvesting. Indeed, it cops with problems such as data dependencies in task graphs, an online variation of solar energy, transient faults, and execution times. It has proposed a flexible schedule at design time and after monitoring the runtime behavior of the system, the lightweight online adjustment mechanism is employed to adapt the task execution policy. The study in [113] has proposed two scheduling algorithms to improve system-level soft-error reliability in conjunction with satisfying lifetime reliability and real-time requirements. If the remaining slack time is sufficient, it can guarantee to recover any faulty task through the dynamic recovery allocation technique. It has satisfied the lifetime reliability requirements by reducing core frequency for appropriate tasks. Hence, aging due to temperature and thermal cycling will be reduced.

F. Combination of Several Fault-Tolerance Techniques

Cia et al. [114] have presented a greedy heuristic-based method to reduce energy consumption in the heterogeneous distributed embedded systems. The proposed method exploits both task replication and re-execution techniques to tolerate faults and satisfies the deadline constraint while reducing energy consumption. In the proposed method, the initialization parts of the tasks are mapped based on the utilization of the cores and then will be executed based on the re-execution policy. Afterward, the greedy algorithm changes the mapping policy and fault-tolerance technique to meet the deadline constraints and reduces energy consumption. The study in [115] has proposed energy-efficient fault-tolerance scheduling for a set of applications with precedence-constrained, which is based on list scheduling heuristics that satisfies the real-time constraints.

V. THERMAL-AWARE FAULT-TOLERANCE TECHNIQUES

Due to the elevated on-chip temperatures, many research efforts have been made to control the power and the temperature of the multi-core chip at the system level, e.g., [124]-[129]. A large body of these techniques has enforced

temperature constraints within the task mapping decisions while aiming at maximizing the performance. For instance, a task mapping technique called DsRM for homogeneous multi-cores is introduced in [130], which selects the numbers and the positions of the active and inactive cores of each multi-threaded application, so that the temperature constrained is satisfied. For heterogeneous multi-cores, the technique proposed in [131] assigns the multi-threaded application to the tiles on the chip, and then maps their threads to the cores, while satisfying the thermally safe power density constraint. To cope with the concern of scalability with the continuous increase in the number of cores, a distributed thermal-constrained task mapping has been proposed in [132]. In addition to task mapping, task migration policies are proposed in [133] to improve the performance under the temperature constraint. However, all of these task mapping techniques do not enforce timing and reliability constraints.

Another class of related works proposes mapping and scheduling techniques that consider both timing and temperature constraints, e.g., [21], [134]-[139]. The study in [134] has proposed two DTM techniques for cyber-physical systems, where the thermal stress is evenly distributed temporally (on each core) and spatially (among cores). In order to balance the thermal related wear-out among cores, tasks will be reassigned among cores with recording their effect on the aging rate. The study in [135] has proposed a task assignment heuristic on heterogeneous platforms to minimize the dynamic energy consumption under timing constraints. The study in [136] estimates the temperature of the multi-core chip while considering the dynamic behavior of the system in real time. Tasks are partitioned and assigned to cores in a way that meets the timing constraints. Then, the proposed peak temperature manager is applied to each task partition based on a given scheduling algorithm. In order to avoid the thermal hotspots on a multi-core chip, the work in [137] has introduced a runtime thermal-aware scheduler based on power-gating and job-migration techniques. The study in [138] minimizes the energy consumption through applying DVS while at the same time meeting thermal constraints. The study in [139] has presented both power-aware and thermal-aware approaches for task allocating and scheduling in embedded systems.

None of these works exploits fault-tolerance techniques to satisfy the reliability target. Just a few works (shown in Table 3) exploit fault-tolerance techniques and at the same time consider all of the three metrics; time, power, and temperature. In the following, these few works will be discussed and classified into sub-categories considering their adopted fault-tolerance techniques, similar to the previous sections.

A. Replication

The task assignment and scheduling technique have been introduced in [116] that uses a mixed-integer linear program (MLP) solver to optimize the makespan under time, reliability, and peak temperature constraints. Makespan is the latest completion time of all tasks on processors. Then a two-stage heuristic is proposed which first determines the task mapping and replication technique to minimize the makespan, and then it checks meeting the timeliness and temperature constraint of the system. Since it has considered the service-oriented systems, all the services (independent tasks) need to be finished before the deadline such that the QoS requirement of users can be satisfied. If the system's temperature constraint is violated, it tries to reduce the peak temperature using two task sequencing and frequency scaling techniques. The effectiveness of the proposed method is evaluated through synthetic random generated tasks set and real-world multimedia applications. The reliability requirement of tasks to tolerate the transient faults is determined in the interval of [0.7, 0.999]. The simulated processor is modeled based on a prototype version of the ARM big.LITTLE chip containing 3 Cortex A7 cores and 2 Cortex A15 cores.

The proposed technique in [117] has optimized energy consumption while satisfying timing, reliability, thermal design power (TDP), and thermal safe power (TSP) [123] constraints through convex optimization. The proposed optimizer maps and schedules periodic hard real-time tasks on different types of cores on heterogeneous multi-core embedded systems. The effectiveness of the optimal solution (YALMIP solver in MATLAB) is evaluated via simulating different task sets of the MiBench Benchmark suite running on ARM Cortex-A7 and Cortex-A15. It is worthy to mention that since the study in [117] exploits optimization, it is applicable for lower number of tasks.

B. Checkpointing (CP)

Zhou et al. [118] have introduced a stochastic energy-efficient thermal- and reliability-aware task scheduling for real-time systems. Instead of exploiting the released slack times for energy reduction, in this paper the generated slack times are utilized for providing fault-tolerance, energy management, and reducing the temperature. First, the slack time is utilized to guarantee a certain level of reliability requirement in the presence of stochastic soft errors by calculating the optimum number of checkpoints for each task. Then it selects the voltage-frequency level of each task based on available slack time to reduce energy consumption. Finally, the temperature is reduced by utilizing the task sequencing technique, and remaining slack times. The thermal-aware task sequencing heuristic method shows that the execution order of a hot task and a cool task has an important effect on the peak temperature. It has been shown that the final temperature of tasks executing in the hot–cool order is less than executing in the order of cool–hot tasks. In order to evaluate the reliability of the generated task schedule under transient fault occurrences of Poisson probability distribution, the Monte Carlo simulation method is utilized. The effectiveness of the proposed method is evaluated via simulating different task sets of the MiBench and Mediabench Benchmark suites running on the ARM Cortex A7 processor.

Table 4. Summary of all fault-tolerant techniques.

Ref.	Design Metrics		
	Time	Power/Energy	Temperature
[38]-[63]	✓		
[64]-[115]	✓	✓	
[116]-[121]	✓	✓	✓

C. N Modular Redundancy (NMR)

In order to conisider temperature constraint within the scheduling process of fault-tolerant mixed-criticality system the study in [119] presents, for the first time, a thermal-aware scheduling scheme, named TherMa-MiCs. In particular, TherMa-MiCs, satisfies the temperature constraint jointly with the timing constraints of the high-criticality tasks, while attempting to maximize the QoS of low-criticality tasks. Moreover, the reliability target is satisfied by employing the N Modular Redundancy (NMR) fault-tolerant technique. The effectiveness of the proposed method is evaluated via simulation of different task sets of the MiBench Benchmark suite running on ARM Cortex-A15.

D. Standby-Sparing

The authors in [120] have proposed a thermal-aware cold standby-sparing technique that maximizes the Quality of Service (QoS) of soft real-time tasks. Their proposed technique tolerates permanent and transient faults for heterogeneous multicore real-time embedded systems while meeting the Thermal Safe Power (TSP) as the core-level power constraint. Executing the main and backup tasks on the cores at any power consumption below TSP guarantees that no thermal violation occurs. Moreover, they have employed a heterogeneous platform to execute the main tasks as soon as possible on high-performance cores with more power budget and the backup tasks are executed on low power cores after finishing the main tasks to maximize the QoS.

E. Combination of Several Fault-Tolerance Techniques

A peak-power-aware reliability management scheme has been proposed in [121] that meets the chip-level power constraints (TPD) and core-level power constraints (TSP) by employing different versions of each soft real-time task (code version programming) and determines the number of required replicas for each task to preserve the system reliability at an acceptable level. It is mentioned that due to the disadvantages of the DVS technique, the Dynamic Frequency Scaling (DFS) technique is exploited to reduce the peak power consumption and satisfy thermal constraints. In order to verify the effectiveness of the proposed method extensive simulations including gem5, McPAT, HotSpot, and TSP tools are exploited. The implementations result in the code versions with low-power and high-reliability consumption. The low-power-density code version of a task is a code that has the lowest peak power and average power consumption among all code versions, while the high-reliability code version of a task is a code that has the highest functional reliability among all versions. The effectiveness of the proposed method is evaluated via simulation of different task sets of the MiBench Benchmark suite running on ARM Cortex-A7, Cortex-A12, and Cortex-A15.

VI. Summary and Future Trends

In this paper, we have presented a survey for many relevant task mapping/scheduling policies for real-time embedded systems that employ fault-tolerance techniques to satisfy reliability requirements. Applying fault-tolerance techniques introduces new challenges to the scheduling process of real-time embedded systems. Firstly, the fault-tolerance techniques come with an additional timing overhead that should be considered in the scheduling process to avoid missing timing constraints. Secondly, fault-tolerance techniques will increase the power consumption of the cores, leading to increasing on-chip temperatures beyond safe limits. Therefore, we have focused on the related works, which consider timing, power/energy, and temperature on single-, dual-, or multi-core processors. Table 4 summarizes the reviewed state-of-the-art task mapping/scheduling policies for fault-tolerance systems according to their considered goals and constraints. As can be seen in this table, several works have considered power and energy besides timing constraints. However, just a few works consider the thermal issue, as well, in spite of its importance. In particular, it is not possible to provide reliability and timing guarantees without enforcing temperature constraints, since chip-level countermeasures will be triggered at any thermal violation, which might lead to violating reliability and timing constraints. Therefore, it is inevitable to jointly consider all of these metrics; reliability, timing, power/energy, and temperature within the task mapping/scheduling process of fault-tolerance embedded systems. Applying multi-objective optimizations for fault-tolerance embedded systems in future research could be a promising approach to exploit optimization potentials for all relevant metrics; time, reliability, power/energy, and temperature.

REFERENCES

[1] H. Esmaeilzadeh, E. Blem, R. St.Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Int'l Conf. on Computer Applications in Industry and Engineering (ISCA)*, 2011, pp. 365–376.

[2] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. When, "Reliable on-chip systems in the nano-era: Lessons learnt and future trends," *50th ACM/EDAC/IEEE Design Automation Conf. (DAC)*, Austin, TX, 2013, pp. 1-10.

[3] J. Henkel, S. Pagani, H. Khdr, F. Kriebel, S. Rehman, and M. Shafique, "Towards performance and reliability-efficient computing in the dark silicon era," *IEEE/ACM Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, Dresden, Germany, Mar 14-18 2016.

[4] R. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction," *Digest. Int'l Electron Devices Meeting*, San Francisco, CA, USA, 2002, pp. 329-332.

[5] R. Canal, C. Hernandez, R. Tornero, A. Cilardo, G. Massari, F. Reghenzani, W. Fornaciari, M. Zapater, D. Atienza, A. Oleksiak, W. Piatek, and J. Abella "Predictive reliability and fault management in

exascale systems: State of the art and perspectives," *ACM Computing Surveys*, vol. 53, no. 5, 2020.

[6] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," *ACM/EDAC/IEEE Design Automation Conf. (DAC)*, 2015.

[7] M. Shafique, S. Garg, D. Marculescu, and J. Henkel, "The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives," *in Proc. ACM/EDAC/IEEE 51st Design Autom. Conf. (DAC)*, 2014, pp. 1-6.

[8] H. Amrouch, H. Khdr, and J. Henkel, Aging effects: From Physics to CAD. In: W. Fornaciari, D. Soudris, (eds) Harnessing Performance Variability in Embedded and High-performance Many/Multi-core Platforms. Springer, Cham, 2019.

[9] J. Henkel, T. Ebi, H. Amrouch, and H. Khdr, "Thermal management for dependable on-chip systems," *18th Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Yokohama, 2013, pp. 113-118.

[10] G. C. Buttazzo, Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, New York, NY, USA: Springer, 2011.

[11] A. Burns, and R. I. Davis, "A survey of research into mixedcriticality systems," *Journal ACM Computing Surveys*, vol. 50, no. 6, 2018.

[12] F. Oszwald, J. Becker, P. Obergfell, and M. Traub, "Dynamic Reconfiguration for Real-Time Automotive Embedded Systems in Fail-Operational Context," *IEEE Int'l Parallel and Distributed Processing Symp. Workshops (IPDPSW)*, Vancouver, BC, Canada, 2018, pp. 206-209.

[13] G. Mehmood, M. Zahid Khan, S. Abbas, M. Faisal, and H. U. Rahman, "An energy-efficient and cooprative fault-tolerant communication approach for wireless body area network," *IEEE Access*, vol. 8, pp. 69134-69147, 2020.

[14] G. Mehmood, M. Z. Khan, A. Waheed, M. Zareei, and E. M. Mohamed, "A trust-based energy-efficient and reliable communication scheme (trust-based ERCS) for remote patient monitoring in wireless body area networks," *IEEE Access*, vol. 8, pp. 131397-131413, 2020.

[15] B. Wang, G. Vakil, Y. Liu, T. Yang, Z. Zhang, C. Gerada, "Optimization and analysis of a high power density and fault tolerant starter–generator for aircraft application" *Energies*, vol. 14, pp 1-113, 2021.

[16] E. Dubrova, "Fault-tolerant design," Springer-Verlag New York, 2013.

[17] I. Koren, and C.M. Krishna, "Fault-tolerant systems," Morgan Kaufman, 2007.

[18] "MiBench homepage." [Online]. Available: http://vhosts.eecs.umich.edu/mibench/. [Accessed: Feb-2021].

[19] Intel Corporation, Santa Clara, CA, USA, Intel Xeon Phi Coprocessor Datasheet, Jun. 2013.

[20] 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," *IEEE Embedded Systems Letters*, 2020.

[21] 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," *IEEE Real-Time Systems Symposium (RTSS)*, Hong Kong, Hong Kong, 2019, pp. 220-232.

[22] R. F. S. 167, *Software considerations in airborne systems and equipment certification, RTCA document DO-178C*. RTCA Incorporated 1992.

[23] T. Li, M. Shafique, J. A. Ambrose, S. Rehman, J. Henkel, and S. Parameswaran, "RASTER: Runtime adaptive spatial/temporal error resiliency for embedded processors," *50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1-7.

[24] J. R. Azambuja, F. Kastensmidt, and J. Becker, "Hybrid fault tolerance techniques to detect transient faults in embedded processors," *Springer Int'l Publishing Switzerland*, 2014.

[25] A. Mahmood, and E. Mccluckey, "Concurrent error detection using watchdog processors-a survey," *IEEE Trans. on Computers (TC)*, vol. 37, no. 2, pp. 160–174, 1988.

[26] T. Austin, and T. Diva, "A reliable substrate for deep submicron microarchitecture design," *in Proc. ACM/ IEEE Int'l Syp. on Microarchitecture, IEEE Computer Society*, 1999, pp. 196–207.

[27] C. Lisboa, M. Erigson, and L. Carro, "System level approaches for mitigation of long duration transient faults in future technologies*," in Proc. IEEE European Test Symp (ETS)*, Los Alamitos, Los Alamitos, USA, 2007, pp. 165-170.

[28] N. Oh, S. Mitra, and E. Mccluckey, "ED4I: error detection by diverse data and duplicated instructions," *IEEE Trans. on Computers (TC)*, vol. 51, no. 2, pp. 180-199, 2002.

[29] P. Cheynet, B. Nicolescu, R. Velazco, M. Rebaudengo, S. Reorda, and M. Violante, "Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors," *IEEE Trans. On Nuclear Science, IEEE Nuclear and Plasma Sciences Society*, vol. 47, no. 6 (part 3), pp. 2231–2236, 2000.

[30] J. Azambuja, A. Lapolli, L. Rosa, and F. Kastensmidt, "Detecting SEEs in microprocessors through a non-intrusive hybrid Technique," *IEEE Trans. On Nuclear Science*, Los Alamitos, vol. 58, no. 3, pp. 993-1000, 2011.

[31] S. Cuenca-Asensi, A. Martinez-Alvarez, F. Restrepo-Calle, F.Palomo, H. Guzman-Miranda, and M. Aguirre, "A novel co-design approach for soft errors mitigation in embedded systems," *IEEE Trans. on Nuclear Science*, Los Alamitos, vol. 58, no. 3, pp. 1059-1065, 2011.

[32] A. Lindoso, L. Entrena, E. Millan, S. Cuenca-Asensi, A. Martinez-Alvarez, and F. Restrepo-Calle, "A co-design approach for SET mitigation in embedded systems," *IEEE Trans. On Nuclear Science*, vol. 59, no. 4, pp. 1034-1039, 2012.

[33] E. Rhod, C. Lisboa, L. Carro, and M. Sonza-reorda, "Hardware and software transparency in the protection of programs against SEUs and SETs," *Journal of Electronic Testing: theory and applications*, vol. 24, no. 3, pp. 45-56, 2008.

[34] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," Lecture Notes, Caltech, Pasadena, CA, USA, January 4-15, 1952. In: Shannon, C.E., McCarthy, J. (eds.): Automata Studies, pp. 43–98. Princeton Univ. Press, Princeton, 1956.

[35] E. F. Moore, and C. E. Shannon, "Reliable circuits using less reliable relays," Part I. J. Frankl. Inst., 262 (3), pp. 191–208, Sep. 1956.

[36] E. F. Moore, and C. E. Shannon, "Reliable circuits using less reliable relays," Part II. J. Frankl. Inst., 262(4): pp. 281–297, Oct. 1956.

[37] D. T. Chiang, and S. Niu. "Reliability of consecutive-k-out-of-n:F system," *IEEE Trans. Reliab.*, R-30(1):87–89, Apr. 1981.

[38] C-C. Han, K. G. Shin, and J. Wu, "A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults," *IEEE Trans. on Comp. (TC)*, vol. 52, no. 3, pp. 362-372, 2003.

[39] J. Kim, K. Lakshmanan, and R. Rajkumar, "R-BATCH: Task partitioning for fault-tolerant multiprocessor real-time systems," *10th IEEE Int'l Conf. on Comp. and Inf. Tec.*, 2010, pp. 1872-1879.

[40] J. Kim, G. Bhatia, R. Rajkumar, and M. Jochim, "SAFER: System-level architecture for failure evasion in real-time applications," *IEEE 33rd Real-Time Systems Symposium (RTSS)*, 2012, pp. 227-236.

[41] P. Dobias, E. Casseau, and O. Sinnen, "Comparison of different methods making use of backup copies for fault-tolerant scheduling on embedded multiprocessor systems," *Conf. on Design and Arch. for Signal and Image Proc. (DASIP)*, Porto, 2018, pp. 100-105.

[42] W. Luo, X. Qin, X. Tan, K. Qin, and A. Manzanares, "Exploiting redundancies to enhance schedulability in fault-tolerant and real-time distributed systems," *IEEE Trans. on Sys., Man, and Cyber. - Part A: Sys. and Hum.*, vol. 39, no. 3, 2009, pp. 626-639.

[43] P. Purushothaman Nair, A. Sarkar, and S. Biswas, "Fault-tolerant real-time fair scheduling on multiprocessor systems with cold-standby," *IEEE Trans. on Dep. and Secure Computing (TDSC)*, 2019.

[44] P. Chevochot, and I. Puaut, "Scheduling fault-tolerant distributed hard real-time tasks independently of the replication strategies," *Proc. 6th Int'l Conf. on Real-Time Computing Systems and Applications (RTCSA)*, Hong Kong, China, 1999, pp. 356-363.

[45] S. Gopalakrishnan and M. Caccamo, "Task partitioning with replication upon heterogeneous multiprocessor systems," *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Jose, CA, USA, 2006, pp. 199-207.

[46] J. Chen, C. Yang, T. Kuo, and S. Tseng, "Real-time task replication for fault tolerance in identical multiprocessor systems," *13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Bellevue, WA, 2007, pp. 249-258.

[47] J. Lin, and A. M. K. Cheng, "Real-time task assignment with replication on multiprocessor platforms," *15th Int'l Conf. on Parallel and Distributed Systems*, Shenzhen, 2009, pp. 399-406.

[48] S. Punnekkat, A. Burns, and R. Davis, "Analysis of checkpointing for real-time systems," *Real-Time Syst.*, vol. 20, no. 1, 2001, pp. 83–102.

[49] H. Tabkhi, S. G. Miremadi, and A. Ejlali, "An asymmetric checkpointing and rollback error recovery scheme for embedded processors," *IEEE Int'l Symposium on Defect and Fault Tolerance of VLSI Systems*, Boston, MA, 2008.

[50] Z. Zhengyong, P. Liping, and Y. Fumin, "Schedulability analysis for Fault tolerance real-time system under fault bursts," *IEEE 7th Int'l Info. Tech. and Artif. Intell. Conf.*, Chongqing, 2014, pp. 20-27.

[51] 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.

[52] F. Restrepo-Calle, S. Cuenca-Asensi, and A. Martinez-Alvarez, "An effective strategy for selective hardening of software," *18th IEEE Latin American Test Symp. (LATS)*, Bogota, 2017, pp. 1-6.

[53] V. Izosimov, P. Pop, P. Eles, and Z. Peng, "Scheduling of fault-tolerant embedded systems with soft and hard timing constraints," *Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, Munich, 2008, pp. 915-920.

[54] B. Zheng, Y. Gao, Q. Zhu, and S. Gupta, "Analysis and optimization of soft error tolerance strategies for real-time systems," *Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Amsterdam, 2015, pp. 55-64.

[55] A. Thekilakkattil, R. Dobrin, S. Punnekkat, and H. Aysan, "Optimizing the fault tolerance capabilities of distributed real-time systems," *Proc. of the 14th IEEE Int'l Conf. on Emerging Technologies and Factory Automation (ETFA)*, Palma de Mallorca, Spain, 2009, pp. 1699–1702.

[56] R. M. Pathan, and J. Jonsson, "Exact fault-tolerant feasibility analysis of fixed-priority real-time tasks," *IEEE 16th Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Macau SAR, 2010, pp. 265-274.

[57] R. M. Pathan, "Fault-tolerant and real-time scheduling for mixed-criticality systems," *Real-Time Syst.*, vol. 50, no. 4, 2014, pp. 509-547.

[58] Z. Al-bayati, J. Caplan, B. H. Meyer, and H. Zeng, "A four-mode model for efficient fault-tolerant mixed-criticality systems," *Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, 2016.

[59] K. Cao, G. Xu, J. Zhou, M. Chen, T. Wei, and K. Li, "Lifetime-aware real-time task scheduling on fault-tolerant mixed-criticality embedded systems," *Future Generation Computer Systems*, 2019, pp. 165-175.

[60] J. Huang, J. O. Blech, A. Raabe, C. Buckl and A. Knoll, "Analysis and optimization of fault-tolerant task scheduling on multiprocessor embedded systems," *Proc. of the 9th IEEE/ACM/IFIP Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Taipei, 2011, pp. 247-256.

[61] 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.

[62] P. Pop, V. Izosimov, P. Eles, and Z. Peng, "Design optimization of time- and cost-constrained fault-tolerant embedded systems with checkpointing and replication," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 3, 2009, pp. 389-402.

[63] J. Caplan, Z. Al-bayati, H. Zeng and B. H. Meyer, "Mapping and scheduling mixed-criticality systems with on-demand redundancy," *IEEE Trans. on Computers (TC)*, vol. 67, no. 4, pp. 582-588, 1 April 2018.

[64] O. S. Unsal, I. Koren and C. M. Krishna, "Towards energy-aware software-based fault tolerance in real-time systems," *Proc. of the Int'l Symp. on Low Power Electronics and Design, Monterey (ISLPED)*, CA, USA, 2002, pp. 124-129.

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

[66] A. Ejlali, B. M. Al-Hashimi, and P. Eles, "Low-energy standby-sparing for hard real-time systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 3, pp. 329-342, 2012.

[67] R. Sridharan, and R. Mahapatra, "Reliability aware power management for dual-processor real-time embedded systems," *Design Automation Conf. (DAC)*, Anaheim, CA, 2010, pp. 819-824.

[68] S. Bansal, R. K. Bansal, and K. Arora, "Energy efficient backup overloading schemes for fault tolerant scheduling of real-time tasks," *Journal of Systems Architecture (JSA)*, vol. 113, 2021.

[69] Y. Guo, D. Zhu, and H. Aydin, "Generalized standby-sparing techniques for energy-efficient fault tolerance in multiprocessor real-time systems," *IEEE 19th Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Taipei, 2013, pp. 62-71.

[70] 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 (JSA)*, vol. 78, pp. 68–80, Aug. 2017.

[71] M. Ansari, S. Safari, F. R.Poursafaei, M. Salehi, and A. Ejlali, "AdDQ: Low-energy hardware replication for real-time systems through adaptive dual-queue scheduling," *The CSI Journal on Computer Science and Engineering*, vol. 15, no. 1, 2017.

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

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

[74] M. Ansari, A. Yeganeh-Khaksar, S. Safari, and A. Ejlali, "Peak-power-aware energy management for periodic real-time applications," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 4, pp. 779-788, 2020.

[75] A. Naghavi, S. Safari and S. Hessabi, "Tolerating permanent faults with low-energy overhead in multicore mixed-criticality systems," *IEEE Trans. on Emerging Topics in Computing (TETC)*, 2021.

[76] S. Safari, S. Hessabi, and G. Ershadi, "LESS-MICS: A low energy standby-sparing scheme for mixed-criticality systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

[77] M. Ansari, M. Salehi, S. Safari, A. Ejlali, and M. Shafique, "Peak-power-aware primary-backup technique for efficient fault-tolerance in multicore embedded systems," *IEEE Access*, vol. 8, pp. 142843-142857, 2020.

[78] I. Assayad, A. Girault, and, H. Kalla, "Scheduling of real-time embedded systems under reliability and power constraints," *IEEE Int'l Conf. on Complex Systems (ICCS)*, Agadir, 2012, pp. 1-6.

[79] F. R. Poursafaei, S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Offline replication and online energy management for hard real-time multicore systems," *Proc. of the 1st Int'l the CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, Tehran, 2015.

[80] J. Spasic, D. Liu, and T. Stefanov, "Energy-efficient mapping of real-time applications on heterogeneous MPSoCs using task replication," *Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Pittsburgh, PA, 2016, pp. 1-10.

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

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

[83] 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 Parallel and Distributed Systems (TPDS)*, vol. 30, no. 10, pp. 2338-2354, 2019.

[84] 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," *Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

[85] A. Yeganeh-Khaksar, M. Ansari, and A. Ejlali, "ReMap: Reliability management of peak-power-aware real-time embedded systems through task replication," *IEEE Trans. on Emerging Topics in Computing (TETC)*, 2020.

[86] E. (Mootaz) Elnozahy, R. Melhem, and D. Mosse, "Energy-efficient duplex and TMR real-time systems," *Proc. of the Real-Time Systems Symposium (RTSS)*, 2002.

[87] D. Zhu, R. Melhem, D. Mosse, and E. Elnozahy, "Analysis of an energy efficient optimistic TMR scheme," *Proc. 10th Int'l Conf. on Parallel and Distributed Systems, (ICPADS)*, Newport Beach, CA, USA, pp. 559-568, 2004.

[88] F. Baharvand, and S.G. Miremadi, "ANMR: Aging-aware adaptive N-modular redundancy for homogeneous multicore embedded processors," *Journal of Parallel and Distributed Computing*, vol. 109, 2017.

[89] F. Baharvand, and S. G Miremadi, "LEXACT: Low energy N-modular redundancy using approximate computing for real-time multicore processors," *IEEE Trans. on Emerging Topics in Computing (TETC)*, vol. 8, no. 2, pp. 431-441, 2020.

[90] S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Energy-budget-aware reliability management in multi-core emebedded systems with hybrid energy source," *The CSI Journal on Computer Science and Engeneering (JCSE)*, vol. 15, no. 2, pp 31-43, 2018.

[91] M. Salehi, A. Ejlali, and B. M. Al-Hashimi, "Two-phase low-energy N-modular redundancy for hard real-time multi-core systems," *IEEE Trans. on Parallel and Distributed Systems (TPDS)*, vol. 27, no. 5, pp. 1497-1510, 1 May 2016.

[92] M. Ansari, S. Safari, A. Yeganeh-Khaksar, M. Salehi, and A. Ejlali, "Peak power management to meet thermal design power in fault-tolerant embedded systems," *IEEE Trans. on Parallel and Distributed Systems (TPDS)*, vol. 30, no. 1, pp. 161-173, 2019.

[93] 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.

[94] Y. Zhang, and K. Chakrabarty, "A unified approach for fault tolerance and dynamic power management in fixed-priority real-time embedded systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, no. 1, pp. 111-125, Jan. 2006.

[95] Y. Zhang, R. Dick, and K. Chakrabarty, "Energy-aware deterministic fault tolerance in distributed real-time embedded systems" *Proc. 41st Design Automation Conf. (DAC)*, 2004.

[96] R. Melhem, D. Mosse, and E. Elnozahy, "The interplay of power management and fault recovery in real-time systems," *IEEE Trans. Computers (TC)*, vol. 53, no. 2, pp. 217-231, 2004.

[97] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," *IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, 2004, pp. 35-40.

[98] T. Wei, P. Mishra, K. Wu, and J. Zhou, "Quasi-static fault-tolerant scheduling schemes for energy-efficient hard real-time systems," *J. of Sys. and Soft.*, vol. 85, no. 6, pp. 1386-1399, 2012.

[99] T. Wei, P. Mishra, K. Wu, and H. Liang, "Fixed-priority allocation and scheduling for energy-efficient fault tolerance in hard real-time multiprocessor systems," *IEEE Trans. on Parallel and Distributed Systems (TPDS)*, vol. 19, no. 11, pp. 1511-1526, Nov. 2008.

[100] D. Zhu, "Reliability-aware dynamic energy management in dependable embedded real-time systems," *ACM Trans. Embed. Comput. Syst.*, vol. 10, no. 2, pp. 1-27, 2010.

[101] Q. Han, M. Fan, and G. Quan, "Energy minimization for fault-tolerant real-time applications on multiprocessor platforms using checkpointing," *Proc. of the Int'l Symposium on Low Power Electronics and Design (ISLPED)*, 2013.

[102] Q. Han, M. Fan, L. Niu, and G. Quan, "Energy minimization for fault-tolerant scheduling of periodic fixed-priority applications on multiprocessor platforms," *Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, Grenoble, 2015, pp. 830-835.

[103] M. Salehi, M. Khavari Tavana, S. Rehman, M. Shafique, A. Ejlali, and J. Henkel, "Two-state checkpointing for energy-efficient fault tolerance in hard real-time systems," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 7, pp. 2426-2437, July 2016.

[104] D. Zhu, and H. Aydin, "Energy management for real-time embedded systems with reliability requirements," *IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD)*, San Jose, CA, 2006, pp. 528-534.

[105] B. Zhao, H. Aydin, and D. Zhu "Generalized reliability-oriented energy management for real-time embedded applications," *Proc. of the Conf. on Computer-Aided Design (ICCAD)*, 2011, pp. 381–386.

[106] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 3, pp. 316-328, 2010.

[107] D. Zhu, and H. Aydin, "Reliability-aware energy management for periodic real-time tasks," *IEEE Trans. on Computers (TC)*, vol. 58, no. 10, pp. 1382-1397, Oct. 2009.

[108] B. Zhao, H. Aydin, and D. Zhu, "Energy management under general task-level reliability constraints," *IEEE 18th Real Time and Embedded Technology and Applications Symposium (RTAS)*, Beijing, 2012, pp. 285-294.

[109] X. Qi, D. Zhu, and H. Aydin, "Global reliability-aware power management for multiprocessor real-time systems," *IEEE 16th Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Macau SAR, 2010, pp. 183-192.

[110] P. Pop, K. H. Poulsen, V. Izosimov, and P. Eles, "Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems," *5th IEEE/ACM/IFIP Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Salzburg, 2007, pp. 233-238.

[111] L. Zhang, K. Li, K. Li, and Y. Xu, "Joint optimization of energy efficiency and system reliability for precedence constraint tasks in heterogeneous systems," *Int'l Journal of Electrical Power & Energy Systems*, vol. 78, pp 499-512, 2016.

[112] Y. Xiang, and S. Pasricha, "Fault-aware application scheduling in low-power embedded systems with energy harvesting," *Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, New Delhi, 2014, pp. 1-10.

[113] Y. Ma, T. Chantem, R. P. Dick, and X. S. Hu, "Improving reliability for real-time systems through dynamic recovery," *Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, Dresden, 2018, pp. 515-520.

[114] Y. Cai, S. M. Reddy, and B. M. Al-Hashimi, "Reducing the energy consumption in fault-tolerant distributed embedded systems with time-constraint," *8th Int'l Symposium on Quality Electronic Design (ISQED)*, San Jose, CA, 2007, pp. 368-373.

[115] B. Kada, and H. Kalla "An efficient fault-tolerant scheduling approach with energy minimization for hard real-time embedded systems," *Distributed Computing for Emerging Smart Networks (DiCES-N). Communications in Computer and Information Science*, vol. 1130. Springer, Cham, 2019.

[116] J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan, and Y. Ma, "Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms," *Journal of Systems and Software (JSS)*, vol. 133, pp. 1-16, 2017.

[117] M. Ansari, M. Pasandideh, J. Saber-Latibari, and A. Ejlali, "Meeting thermal safe power in fault-tolerant heterogeneous embedded systems," *IEEE Embedded Systems Letters*, vol. 12, no. 1, pp. 29-32, March 2020.

[118] J. Zhou, and, T. Wei, "Stochastic thermal-aware real-time task scheduling with considerations of soft errors," *Journal of Systems and Software (JSS)*, vol. 102, pp. 123-133, 2015.

[119] S. Safari, H. Khdr, P. Gohari-Nazari, M. Ansari, S. Hessabi, and J. Henkel, "TherMa-MiCs: Thermal-Aware Scheduling for Fault-Tolerant Mixed-Criticality Systems," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2021.

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

[121] 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 Parallel and Distr. Sys. (TPDS)*, vol. 31, no. 3, pp. 623-633, 2019.

[122] S. Pagani, H. Khdr, W. Munawar, J-J Chen, M. Shafique, M. Li, and J. Henkel, "TSP: Thermal safe power - Efficient power budgeting for many-core systems in dark silicon," *Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, New Delhi, India, Oct 12-17 2014.

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

[124] H. Khdr, H. Amrouch, and J. Henkel, "Aging-constrained performance optimization for multi cores," *ACM/ESDA/IEEE Design Automation Conf. (DAC)*, 2018, pp. 1-6.

[125] S. Pagani, S. Manoj, P. D. Axel Jantsch, and J. Henkel, "Machine learning for power, energy, and thermal management on multicore processors: A survey," *IEEE Trans. on CAD of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 1, Jan 2020.

[126] W. Munawar, H. Khdr, S. Pagani, M. Shafique, J-J. Chen, and J. Henkel, "Peak power management for scheduling real-time tasks on heterogeneous many-core systems," *20th IEEE Int'l Conf. on Parallel and Distributed Systems (ICPADS)*, Hsinchu, Taiwan, 2014.

[127] H. Khdr, H. Amrouch, and J. Henkel, "Dynamic guardband selection: Thermal-aware optimization for unreliable multi-core systems," *IEEE Trans. on Computers (TC)*, Jan 2019.

[128] J. Henkel, H. Khdr, and M. Rapp, "Smart thermal management for heterogeneous multicores (Special Session)," *IEEE/ACM 22nd Design, Automation and Test in Europe Conf. and Exhibition (DATE), Florence, Italy,* Mar 25-29 2019.

[129] A. K. Singh, S. Dey, K. R. Basireddy, K. McDonald-Maier, G. V. Merrett, and B. M. Al-Hashimi, "Dynamic energy and thermal management of multi-core mobile platforms: A survey," *IEEE Design & Test*, 2020.

[130] H. Khdr, S. Pagani, M. Shafique, and J. Henkel, "Thermal constrained resource management for mixed ILP-TLP workloads in dark silicon chips," *ACM/EDAC/IEEE 52nd Design Automation Conf. (DAC), San Francisco, CA, USA*, Jun 7-11 2015.

[131] H. Khdr, S. Pagani, É. 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 (TC)*, vol. 66, no. 3, Mar 2017.

[132] H. Khdr, M. Shafique, S. Pagani, A. Herkersdorf, and J. Henkel, "Combinatorial auctions for temperature-constrained resource management in manycores," *IEEE Trans. on Parallel and Distributed Systems (TPDS)*, Jul 2020.

[133] M. Rapp, M. Sagi, A. Pathania, A. Herkersdorf, and J. Henkel, "Power- and cache-aware task mapping with dynamic power budgeting for many-cores," *IEEE Trans. on Computers (TC)*, vol. 69, no 1, Jan 2020.

[134] S. Xu, I. Koren, and C. M. Krishna, "Thermal aware task scheduling for enhanced cyber-physical systems sustainability," *IEEE Trans. on Sustainable Computing*, vol. 5, no. 4, pp. 581-593, 2020.

[135] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 35, no. 8, pp. 1269-1282, Aug. 2016.

[136] B. Yun, K. G. Shin, and S. Wang, "Predicting thermal behavior for temperature management in time-critical multicore systems," *IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Philadelphia, PA, 2013, pp. 185-194.

[137] B. Yun, K. G. Shin, and S. Wang, "Thermal-aware scheduling of critical applications using job migration and power-gating on multi-core chips," *IEEE 10th Int'l Conf. on Trust, Security, and Privacy in Computing and Communications*, Changsha, 2011, pp. 1083-1090.

[138] S. Wang, J. Chen, Z. Shi, and L. Thiele, "Energy-efficient speed scheduling for real-time tasks under thermal constraints," *15th IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Beijing, 2009, pp. 201-209.

[139] W. Hung, Y. Xie, N. ViJ'aykrishnan, M. Kandemir, and M. J. Irwin, "Thermal-aware task allocation and scheduling for embedded systems," *Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, Munich, Germany, 2005, pp. 898-899.

SEPIDEH SAFARI received the M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2016 with an excellent grade and the first rank. She received the Ph.D. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2021. She was a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany, from 2019 to 2021. Her research interests include 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.

MOHSEN ANSARI received his M.Sc. and Ph.D. degrees in computer engineering from Sharif University of Technology, Tehran, Iran, in 2016 and 2021, respectively. He was a visiting researcher in the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany, from 2019 to 2021. Now, he is a postdoctoral researcher and a research group leader of Embedded Systems Research Laboratory (ESR-LAB), and a lecturer at the department of computer science and engineering, Sharif University of Technology. His research interests include low-power design of embedded systems and multi-/many-core systems with a focus on dependability/reliability.

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 in 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.

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 Sharif University of Technology, Tehran, Iran. His research interests are thermal management in multi-/many-core systems and design of embedded systems with a focus on low-power and 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 System Research Laboratory (ESR-LAB) at

the department of computer engineering at Sharif University of Technology. His research interests are embedded system design, low power system design, fault-tolerant system design, and computer architecture.

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

SHAAHIN HESSABI received the BS and MS degrees in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1986 and 1990, respectively, and the PhD degree in electrical and computer engineering from the University of Waterloo, Ontario, Canada. He joined Sharif University of Technology, in 1996. Since 2007, he has been an associate professor in the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. He has published more than 100 refereed papers in the related areas. His research interests include cyber-physical systems, reconfigurable and heterogeneous architectures, network-on-chip, and system-on-chip. He has served as the program chair, general chair, and program committee member of various conferences, like DATE, NOCS, NoCArch, and CADS.

ALIREZA EJLALI is an Associate Professor of Computer Engineering at Sharif University of Technology, Tehran, Iran. He received a Ph.D. degree in computer engineering from Sharif University of Technology in 2006. From 2005 to 2006, he was a visiting researcher in the Electronic Systems Design Group, University of Southampton, UK. In 2006 he joined Sharif University of Technology as a faculty member in the department of computer engineering and from 2011 to 2015, he was the director of Computer Architecture Group in this department. He is now the director of Embedded Systems Research Laboratory (ESR-LAB) and the head of the department of computer engineering, Sharif University of Technology. His research interests include low power design, fault tolerance, real-time embedded systems, and Internet of Things (IoT).

JÖRG HENKEL (M'95-SM'01-F'15) is currently with the Karlsruhe Institute of Technology (KIT), Germany, where he is directing the Chair for Embedded Systems (CES). Prof. Henkel received the masters and the Ph.D. (Summa cum laude) degrees, both from the Technical University of Braunschweig, Germany. He then joined the NEC Laboratories, Princeton, NJ, USA. His current research interests include design and architectures for embedded systems with focus on low power and reliability. Prof. Henkel has received the 2008 DATE Best Paper Award, the 2009 IEEE/ACM William J. Mc Calla ICCAD Best Paper Award, the CODES+ISSS 2011, 2014 and 2015 Best Paper Awards. He was the General Chair of major CAD events incl. ICCAD and ESWeek. He is the Chairman of the IEEE Computer Society, Germany Section, and was the Editor-in-Chief of the ACM Transactions on Embedded Computing Systems for two terms. He is currently the Editor-in-Chief of the IEEE Design&Test Magazine. He is also an Initiator and Spokesperson of the national priority program on Dependable Embedded Systems of the German Science Foundation and the site coordinator (Karlsruhe site) of the three-university collaborative research center on invasive computing. He is a Fellow of the IEEE and holds ten US patents.