IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 25, 2020, accepted October 8, 2020, date of publication October 14, 2020, date of current version October 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3031039

FANTOM: Fault Tolerant Task-Drop Aware Scheduling for Mixed-Criticality Systems

BEHNAZ RANJBAR[1,2], BARDIA SAFAEI[2,3], (Student Member, IEEE), ALIREZA EJLALI[2], AND AKASH KUMAR[1], (Senior Member, IEEE)

[1]CFAED, Chair for Processor Design, Technische Universität (TU) Dresden, 01069 Dresden, Germany
[2]Department of Computer Engineering, Sharif University of Technology, Tehran 11365-11155, Iran
[3]Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany

Corresponding author: Akash Kumar (akash.kumar@tu-dresden.de)

ABSTRACT Mixed-Criticality (MC) systems have emerged as an effective solution in various industries, where multiple tasks with various real-time and safety requirements (different levels of criticality) are integrated onto a common hardware platform. In these systems, a fault may occur due to different reasons, e.g., hardware defects, software errors or the arrival of unexpected events. In order to tolerate faults in MC systems, the re-execution technique is typically employed, which may lead to overrun of high-criticality tasks (HCTs), which necessitates the drop of low-criticality tasks (LCTs) or degrading their quality. However, frequent drops or relatively long execution times of LCTs (especially mission-critical tasks) are not always desirable and it may impose a negative impact on the performance, or the functionality of MC systems. In this regard, this article proposes a realistic MC task model and develops a design-time task-drop aware schedulability analysis based on the Earliest Deadline First with Virtual Deadline (EDF-VD) algorithm. According to this analysis and the proposed scheduling policy based on the new MC task model, in the high-criticality (HI) mode, when an HCT overruns and the system switches to the HI mode, the number of drops per LCT is prohibited from passing a predefined threshold. In addition, to guarantee the real-time constraints and safety requirements of MC tasks in the presence of faults (assuming transient faults in this article), a corresponding scheduling mechanism has been developed. According to the obtained results from an extensive set of simulations, which have been validated through a realistic avionic application, the proposed method improves the acceptance ratio by up to 43.9% compared to state-of-the-art.

INDEX TERMS Mixed-criticality system, fault-tolerance, mission-critical tasks, drop-aware schedulability test, scheduling policy.

I. INTRODUCTION

Mixed-Criticality (MC) systems are getting more attention due to their broad range of applications in various industries, e.g., medical devices and avionics [1], [2]. In these systems, based on the integration of different tasks with different types of deadlines, safety and certification requirements, a level of criticality is assigned to every MC task [2]–[5]. In this regard, a set of industrial standards, e.g., DO-178B [6], has been introduced with five levels of safety, i.e., *A*, *B*, *C*, *D*, and *E*, (A and E provide the highest and the lowest levels of safety, respectively), which is illustrated in Table 1 [3], [7], [8]. As shown in this table, occurrence of a failure in tasks with various criticality levels has a different impact on the system [9], [10]. To guarantee the system's safety, the Probability-of-Failure-per-Hour (PFH) (which is adopted by safety standards) is determined for all the criticality levels [8], [11]. These MC tasks from different criticality levels are executed on common hardware platforms, which could be single or multi-core. Avionic systems are an example of single-core MC systems, in which tasks are executed on a single-core processor [3], [8], [12].

From an MC system operational perspective, these systems can operate in different criticality modes. Initially, the system operates in the low-criticality (LO) mode, in which all the tasks must be executed before their deadlines. However, if, for instance, a failure has happened in High-Criticality Tasks (HCTs) and the execution time of at least

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

TABLE 1. DO-178B safety requirement.

x	A	B	C	D	E
PFH_x	$< 10^{-9}$	$< 10^{-7}$	$< 10^{-5}$	$> 10^{-5}$	-
Failure Condition	Catastrophic	Hazardous	Major	Minor	No Effect

one HCT exceeds its Worst-Case Execution Time (WCET), these tasks will overrun and the system switches to the high-criticality (HI) mode. In this situation, if we do not drop or degrade Low-Criticality Tasks (LCTs), the deadline of HCTs (such as safety-critical tasks) may be missed, which can lead to catastrophic consequences [2]–[5], [8], [13]. Nevertheless, the frequent deadline misses or service degradation of some LCTs, such as mission-critical tasks (MCTs), may have a negative impact on the other HCTs and MCTs themselves, and consequently on the entire system and may prevent the system from accomplishing its mission correctly. Hence, in the HI mode, limiting the number of frequent drops per LCT is a big challenge.

For instance, consider an MC system whose mission is to capture images in specific time intervals. In this application, the engine operation (i.e., the function that ensures the safe execution of the operation) and the operation of capturing images are considered as the HCTs and LCTs, respectively [5], [14]. Accordingly, if the system switches to the HI mode, due to the execution of HCTs, the existing scheduling methods may frequently drop LCTs or suspend them for a long time, which is not acceptable for a system whose main mission is capturing images. Therefore, the number of allowable drops for every LCT must be restricted in such MC systems. On the other hand, to guarantee the safety and correctness of executing MC tasks before their deadlines, it is crucial to exploit fault-tolerance techniques in designing MC systems [15]. Therefore, scheduling MC tasks by realizing reliability considerations is one of the major challenges in satisfying real-time system requirements in the presence of faults [7], [14], [16].

There have been many studies on MC systems, which have focused on the feasibility of schedules and meeting the timing constraints in both HI and LO modes. In order to guarantee the correct execution of HCTs before their deadlines in the HI mode, previous studies have provided techniques, where they discard LCTs [4], [10], [17]–[21] or degrade them [2], [22]–[28]. In this regard, to degrade LCTs, two approaches have been provided: 1) Decreasing the LCT's WCET, and 2) Increasing the LCT's period. However, none of these algorithms can be applied to MC tasks, because they should not be frequently dropped or postponed for a long time [1], [3], [5], [12]. On the other hand, some papers present priority-based scheduling algorithms for non-MC systems in which tasks have priority [29], [30]. Although the researchers have proposed methods to not drop tasks with lower priorities frequently, but they cannot guarantee the safety requirement of the tasks in safety-critical systems. In addition, they have considered one WCET for each task that wastes the processor capacity (i.e., space) and cost, which are not acceptable for today's embedded systems [1]. Therefore, heuristics in these papers cannot be applied to MC systems.

Inspired by the recent works, in this article, we have considered fault-tolerant MC embedded systems, in which tasks are HCTs or LCTs. MC tasks can be safety-critical, mission-critical or non mission-critical. Since safety-critical tasks are vital and their failure has a more devastating effect than mission-critical ones, we consider safety-critical tasks as HCTs, while mission-critical and non-critical tasks as LCTs. It is worth mentioning that in the LO mode, all types of tasks have to be executed unobstructedly. In case of exceeding the WCET of HCTs, the system switches to the HI mode, where the deadline of all HCTs must be met under any circumstances to avoid catastrophic consequences. In this regard, LCTs may be dropped, while some LCTs, i.e. MCTs, should not be frequently dropped. In addition, service degradation is not acceptable for this type of tasks.

To resolve this issue, we have proposed FANTOM (**FA**ult Tolera**N**t **T**ask-Dr**O**p Aware Scheduling For **M**ixed-Criticality Systems), a novel technique, which is based on a new MC task model and scheduling analysis of MC tasks with different criticality levels, low-critical and high-critical, by considering safety requirements. In FANTOM, the schedulability analysis is conducted in an off-line manner in order to guarantee that all tasks with different criticality levels are executed properly before their deadlines in the presence of faults and based on the operational mode of MC systems. Thus, the main objective of FANTOM is to execute the majority of the LCTs in the HI mode by considering a maximum allowable number of drops for every LCT. In addition, we guarantee the safety requirement of all MC tasks in both LO and HI modes. This is despite the fact that most of the related works are not able to guarantee it in the HI modes. Furthermore, the proposed method can schedule more task sets (i.e., it has a higher acceptance ratio) as compared to similar works [8]. In summary, the main contributions of this work are:

- A new MC task model in which LCTs are dropped consciously in the HI mode (i.e., by introducing a maximum allowable number of drops for every LCT).
- A novel technique (FANTOM) based on the proposed MC task model and the scheduling policy in the HI mode, in which an MC task schedulability analysis is developed by considering safety requirements and fault tolerance.

To the best of our knowledge, FANTOM is the first study of its kind, which considers the scheduling analysis of MC tasks in order to prevent frequent drops of LCTs in the HI mode by assigning a pre-defined threshold to them, while the safety requirements of tasks are guaranteed.

The rest of this article is organized as follows. In Section II, we review the related works. In Section III, the models and assumptions, which have been considered are discussed. The problem statement and motivational example are presented in Section IV. In Section V, we describe FANTOM

TABLE 2. Summary of state-of-the-art approaches.

#		Safety Req. Guarantee	Fault Tolerance	Priority-based/ MC system	LCTs' Drop-Aware in the HI mode
1	[4], [7], [10], [12], [17]	×	×	MC	× (drop all LCTs)
2	[18]–[21]	×	×	MC	× (drop freq.)
3	[2], [22]–[28]	×	×	MC	× (degrade)
4	[31]–[35]	×	✓	MC	× (drop all LCTs)
5	[15], [36]	✓	✓	MC	× (drop freq.)
6	[8], [11]	✓	✓	MC	× (degrade. or drop all)
7	[29], [30]	×	×	Priority-based	- -
8	Proposed Method	✓	✓	MC	✓ (drop-aware)

in detail, while our experimental results have been described in Section VI. Finally, we conclude our paper in Section VII.

II. RELATED WORKS

Many studies in the context of MC systems have only focused on proposing techniques in the field of task scheduling in different operational modes. Since our focus is on single-core processors, we only consider the works presented for MC with a similar scope. Table 2 summarizes the recent works with different targets. As can be seen, a number of works have focused on the feasibility of schedules and meeting the timing constraints in both HI and LO modes (row 1-3). Some of the proposed algorithms are based on the Earliest Deadline First with Virtual Deadline *(EDF-VD)* [4], [5], [7], [10], [20], [24], [28], Early-Release EDF *(ER-EDF)* [2], [22] or Fixed-Priority *(FP)* [17], [31]. A large number of papers of this field and their algorithms have been reviewed in [1]. Most of the existing MC scheduling algorithms discard LCTs when the system switches to the HI mode (shown in row 1). This operation causes serious service interruptions for LCTs. Row 2 shows the papers that try to increase the QoS of LCTs in the HI mode [18]–[21]. Researchers in [18], [21] have concentrated on MC systems that they try to drop a specific number of LCTs in the HI mode. However, they do not guarantee to not drop LCTs frequently. In addition, the recent studies [2], [22]–[28], have provided techniques to improve the minimum service level of LCTs in the HI mode by reducing the WCET of LCTs in the LO mode or increasing their period in the HI mode (row 3). Indeed, they degrade the service level of LCTs that the minimum service level would be guaranteed by their techniques. However, the common part of all of the previous methods is their consideration on an MC model in which LCTs are dropped or degraded when the system switches to the HI mode. Thus, none of these algorithms can be applied to MC tasks that LCTs could not be frequently dropped or postponed for a long time.

Some research works have addressed both fault tolerance and scheduling analysis in MC systems, as shown in rows 4-6. In these papers, some techniques, e.g., re-execution, check-pointing and replication are used to tolerate faults. The methods in [32]–[36] have used Vestal task model [12], in which all LCTs are dropped in the HI mode (row 4). Hence, they did not guarantee the safety requirement and they just improve the reliability. Besides, a few papers [15], [37] have improved QoS of LCTs in the HI mode, while considered fault-tolerance and safety requirement (row 5). Although Caplan *et al.* in [37] have tried to increase QoS of LCTs in the HI mode or when a fault occurs in HCTs in the LO mode, they may drop LCTs in the LO mode and also in the HI mode frequently, which is not acceptable, especially for MCTs. On the other hand, researchers in [8], [11] estimate the WCET of each HCT in each criticality level by obtaining the number of re-executions to guarantee safety requirements; their method drops or degrades LCTs in the HI mode (row 6). Although their method of estimating WCET in each criticality level is used in our paper, degrading or dropping LCTs in a frequent manner without any restrictions in the HI mode (which was used in all previous works) is not desirable and may negatively affect the safety and even leading into catastrophic consequences.

From the perspective of fault-tolerant control of the systems, in general since the fault occurrence is common in many applications, reliability management becomes important [38]. Besides, applying the fault-tolerance techniques may limit a system implementation and degrade its performance. In general, previous research works in the field of fault-tolerant control can be classified into two categories – passive design approaches and active design approaches [39], [40]. Although active approaches increase the performance of a system, they have some limitations and may not control the safety-critical systems in a certain period of time, at runtime [39]. Therefore, since we target the MC systems and performance degradation for HCTs are not admissible, passive design approaches are more appropriate for MC systems. Hence, in passive approaches, the system is designed to achieve the acceptable performance for the system in both non-faulty and faulty behaviors [40].

Some research works [29], [30] have presented scheduling algorithms for priority-based systems, which are non-MC systems (row 7). These algorithms try to execute all high-priority tasks and most low-priority tasks. However, these methods that execute most of the lower priority tasks, are not suitable for systems with safety-critical tasks (i.e., which have high-priority) and MCTs (i.e., which have lower priority), due to the possibility of frequent drops in MCTs. In addition, all these papers have considered one WCET for each task. The WCETs are calculated by different tools, which can be optimistic or pessimistic. If optimistic WCET is considered for the tasks, the deadline of the tasks with higher priorities may be missed and

a catastrophic consequence may happen. If they consider pessimistic WCETs, the processor capacity usage, cost and power would be wasted, which is not acceptable in today's embedded systems. Therefore, these algorithms cannot be applied to MC systems.

In this work, we propose a realistic MC task model and analyze its schedulability based on the EDF-VD algorithm. In this model, some LCTs (i.e., MCTs) cannot be frequently dropped, and service degradation is not acceptable for these tasks. Further, the safety requirements of all types of tasks are guaranteed in both operational modes, LO and HI.

III. SYSTEM MODEL AND DEFINITION

In this section, we describe our system model. At first, we present our MC task model, the system operation during runtime, and we outline the fault model and the fault-tolerance technique. Then, we explain the scheduling algorithm used in this article. As we used several symbols and notations, Table 3 provides a list of them, used in this article, for easy following.

A. TASK MODEL

In this article, we have considered a task set, which consists of n independent periodic tasks running on a single-core processor. Since some of the MC systems require a high level of safety due to their timing requirements [41], we have exploited criticality levels similar to what was defined in [3], in which each criticality level has a requirement based on the deadline and safety requirements. In this regard, two criticality levels have been considered: 1) high, and 2) low. A portion of the tasks are HCTs, which could be considered as the safety-critical tasks, while the other tasks are LCTs, which can be classified to a lower level as mission-critical or non-mission-critical. In our system model, every task (τ_i) is defined as:

$$\tau_i = (\zeta_i, C_i, T_i, d_i, \delta_i) \tag{1}$$

where ζ_i denotes the criticality level of the tasks (i.e., high-critical or low-critical). T_i and C_i denote the period and WCET of the task τ_i. The deadline of a task (d_i) is equal to its period [42]. According to the criticism of an MC system formal model, which is mentioned in [16], and [41], dropping LCTs (except for non-criticality tasks) in favor of HCTs is not a suitable protection mechanism in industrial applications. Hence, dropping LCTs (except for non-criticality tasks) is not permitted in industrial applications, but depending on the type of the application, some of the LCTs (i.e., MCTs) could be dropped. In this regard, frequent dropping or postponing their execution for a long time in the HI mode is not appropriate. Thus, we have introduced and assigned a new parameter δ for every task, which limits the minimum interval between two consecutive drops, that is set to ($\delta_i \times T_i$). Since dropping HCTs is prohibited, we have set $\delta_{HCT} = \infty$, which means that no dropping is allowed for HCTs. In addition, some LCTs are non-critical or non-mission critical, and dropped in the HI mode. Therefore, we define $\delta = 1$ for these tasks. Based on the application, the value of δ for each HCT is determined by designer.

TABLE 3. A list of used notations used in this article.

Symbol	Description
ζ_i	Criticality level of task τ_i
C_i	WCET of task τ_i
T_i	Period of task τ_i
d_i	Deadline of task τ_i
$\hat{d}_i$	Virtual deadline of task τ_i
δ_i	Skip parameter of task τ_i
n_{LCT}	max. required re-execution times for LCTs in the LO mode
n_{HCT}	max. required re-execution times for HCTs in the HI mode
n'_{HCT}	max. required re-execution times for HCTs in the LO mode
f_i	probability factor of an unsuccessful execution of task τ_i
x	multiplying factor to derive virtual deadline
HP	Hyper period of task set
HP'	Hyper period of task set in the unit of hour
C_i^{LO}	WCET of task τ_i in the LO mode
C_i^{HI}	WCET of task τ_i in the HI mode

B. INDUSTRIAL SAFETY STANDARD DEFINITION

There are various safety standards used in industries, such as DO-178B [6] for avionics, and ISO 26262 [43] for road vehicles. These standards define different levels of safety for functions, called Safety Integrity Level (SIL) for automotive domains and Design Assurance Level (DAL) for avionics domain [16], [41]. As mentioned in Section I in detail, five levels are defined in DAL, *A*, *B*, *C*, *D*, and *E*, that A and E provide the highest and the lowest levels of safety, shown in Table 1. Besides, SIL is introduced in four levels in which SIL-1 has the lowest level and SIL-4 has the highest level. The ability of avoiding harm or damage is more crucial in higher SIL/DAL. In terms of the condition upon failure in different levels, SIL-4 can be considered equivalent to DAL-A, and correspondingly, SIL-1 is equivalent to DAL-D [16]. In this article, we target avionic applications with the defined safety levels of DAL and the PFH value for each level.

C. FAULT MODEL, FAULT-TOLERANCE AND SAFETY REQUIREMENTS

Transient faults are the most common faults in embedded systems [35], [44], [45]. If a fault occurs, it may lead to exceeding of the WCET (C_i) and if an HCT is not executed correctly before its deadline due to the fault, it may create catastrophic consequences [8], [16], [41]. Accordingly, we have considered a probability factor (f_i) (Probability of Failure (PoF)), which indicates the probability of an unsuccessful execution of a task due to transient hardware/software faults [46]. Hence, occurrence of a fault in a system is independent of the criticality levels of tasks or criticality modes of the system. Indeed, a fault occurs due to the hardware component defects, electromagnetic interference, etc. [39], [46], [47]. In addition, the PFH has been exploited in this article to measure the safety of the system. The PFH represents the rate of the average system failures in an hour [8], [11], [46]. According to safety standards, PFH estimates the

failure probability of safety functions in each of the criticality levels [8], [46]. As shown in Table 1, five criticality levels of the exploited DO-178B safety standard, i.e., *A, B, C, D* and *E* have been illustrated and the PFH values for all of these levels have been determined. Since the re-execution is one of the most common methods used for fault tolerance [48], in our system model, we have used re-execution of the tasks to tolerate faults and improve the system's reliability according to Table 1. The number of re-executions for each criticality level of tasks, high and low, in each mode to guarantee safety requirement is obtained by using the value of PFHs, which will be discussed later in Section V. In this regard, to probe the occurrence of transient faults, after the execution of each task, its correctness will be checked by an error detection mechanism [46], which its time overhead is added to the task's WCET. Therefore, it will be considered as part of the WCET.

D. MC TASKS ANALYSIS

In this section, we analyze the MC tasks from the WCET perspective. As previously mentioned, every LCT may be mission-critical or even it could have lower safety levels than MCTs. For the last one, these tasks are similar to soft real-time tasks, which their deadline misses may be acceptable and has less or no impact on the safety [3], [5], [14]. On the other hand, the well-known Vestal task model considers several WCETs for HCTs according to the mode of the system. Researchers in [16] and [41] have rejected this model due to the importance of MC industrial applications in which more than one WCET is not acceptable. Indeed, two estimations of WCETs (pessimistic and realistic) in the Vestal task model is not practical in industrial standards. Therefore, in order to estimate WCET, we have exploited the proposed mechanism in [8]. In this task model, one WCET is estimated and then, the WCET of each task in each criticality level will be obtained according to the safety standard. In addition, the re-execution of the tasks has been used in order to tolerate transient faults. Accordingly, this technique will have a direct impact on the execution time of the tasks. Consequently, the number of re-executions for every task to guarantee its safety requirement would lead into specified WCET for each criticality level [8]. Hence, any of the jobs in both LCTs and HCTs, requires maximum n_ζ times (ζ = LCT or HCT) to be executed with regards to safety requirements ($C_i^\zeta = n_\zeta \times C_i$) [8]. In the worst-case of executing all n_ζ times, it may cause the system to be overloaded. Therefore, n'_{HCT} where $n'_{HCT} < n_{HCT}$, has been defined for HCTs, to give the system this ability to switch to the HI mode in case of not having the correct response ready after executing any HCT for n'_{HCT} times ($C_i^{LO} = n'_{HCT} \times C_i, \zeta_i = HCT$). In addition, when the system switches to the HI mode, LCTs may be dropped in order to pave the way for HCTs to be correctly executed in this mode until the system switches back to the LO mode in a safe manner. The details of computing the WCET for each mode operation will be explained in Section V-A.

E. MC SYSTEM OPERATION

Regarding the operation of an MC system, at the first stage, the system begins its operation in the LO mode and all the tasks will be executed by their WCET in the LO mode. As discussed in previous sections, in this LO mode, all tasks (HCTs and LCTs) are scheduled with their specified number of re-execution, which is obtained by using the value of their PFH level. If any of the HCTs overrun their specified time in the LO mode (for example, due to occurrence of the faults), the system will switch to the HI mode. In this case, while the safety and the schedulability of the system are preserved, all HCTs and most of the LCTs (according to our policy) will be executed by their WCET in the HI mode and the non-criticality tasks will be dropped. Hence, in this HI mode, HCTs are planned to schedule with their maximum number of re-execution, obtained based on their PFH. The system remains in the HI mode until all the ready HCTs in the queue have been executed in the core. Afterward, the system can be safely switched back to the LO mode.

F. EDF-VD ALGORITHM FOR MC TASK SCHEDULING

To schedule MC tasks in the single-core processor, we need a scheduling algorithm. We use the existing MC scheduling technique, EDF-VD. The complete analysis of the EDF-VD scheduling algorithm was presented in [14] for the first time. The authors have considered a dual-criticality system, which defines two levels of criticality for its tasks. The main contribution of that paper is that it has decreased the deadlines of HCTs by multiplying the actual deadline by x ($0 < x < 1$). The resulting deadline is called a virtual deadline. This policy would provide a higher priority for HCTs in the scheduling algorithm. When the system is in the LO mode, the virtual deadlines will be used for HCTs in the EDF scheduler and also all of the HCTs will be executed before their deadlines. Nevertheless, when the system switches to the HI mode, the actual deadlines of HCTs will be used in the EDF scheduler and all of the LCTs will be dropped. In this article, we apply the EDF-VD algorithm to the task set in a single-processor. The task scheduling has been done according to our policy. An appropriate interval of x and the required conditions for the EDF-VD algorithm for scheduling a given set of MC tasks are presented in Section V-C.

IV. PROBLEM OBJECTIVES AND MOTIVATION

In this article, we propose a new MC task model and provide a task-drop aware scheduling analysis for it. Four objectives have been set to be achieved in this article: 1) All of the MC tasks should be executed by their deadlines in the LO mode, 2) In cases that the system switches to the HI mode, all of the HCTs should be finished before their specified deadline, 3) It should be guaranteed that the MCTs should not be frequently dropped (indeed, they should be executed before their deadlines by defining a new parameter in the HI mode, which limits their number of drops), and 4) The non MCTs (also known as non-criticality tasks) will be dropped in the

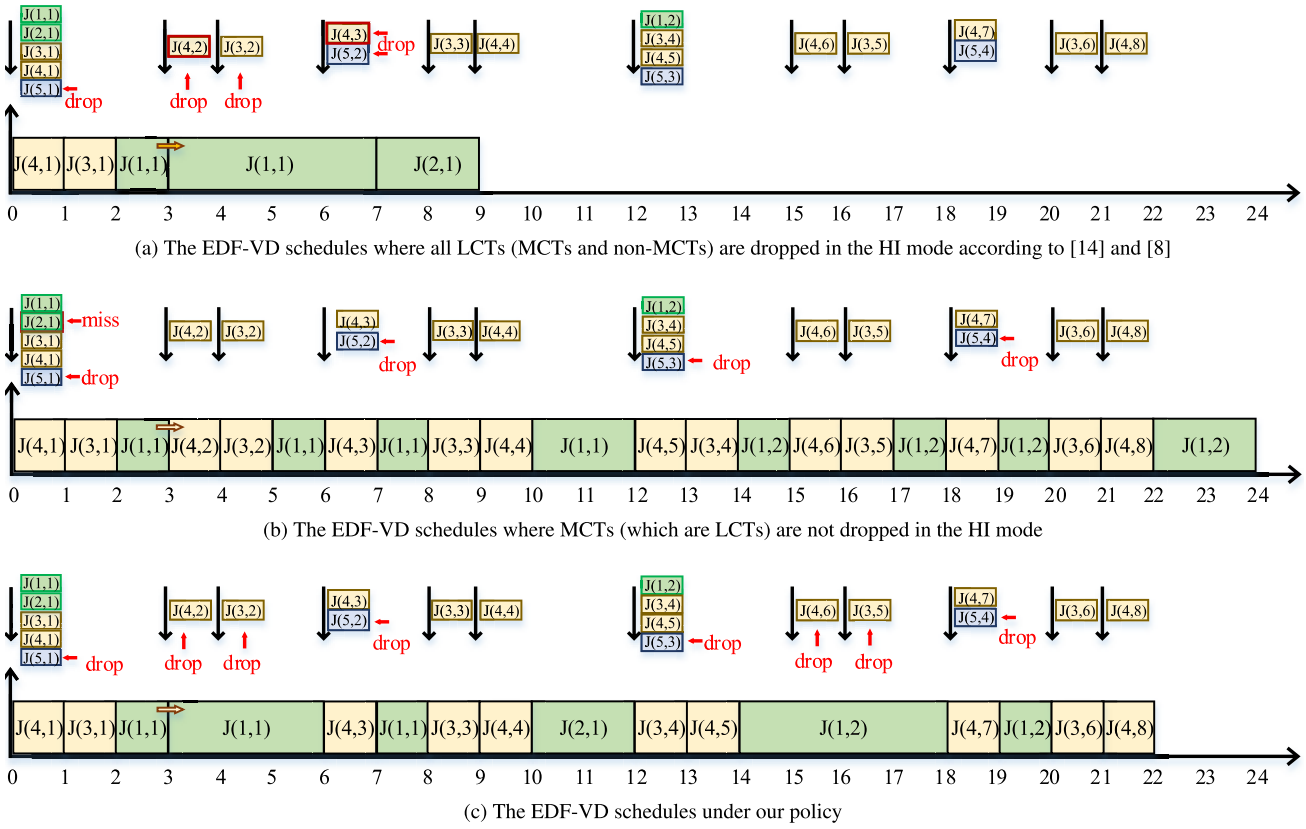

FIGURE 1. Different schedules for the MC task set example within the interval [0, 24].

HI mode due to the deadline meeting of HCTs and MCTs. In the following, a motivational example will be discussed in which a task set containing all types of tasks has been considered.

A. MOTIVATIONAL EXAMPLE

In this section, we are going to give a motivational example based on Fig. 1 to clarify the problem and our solution for limiting the number of frequent drops per LCT. In this regard, assume that a single-core executes five MC tasks ($\tau_1, \tau_2, \tau_3, \tau_4, \tau_5$). The timing parameters of each task are shown in Table 4. As mentioned in Section III-A, the deadlines of the tasks are set to their periods. Each instance (job) of a task τ_i must be executed in the task time period and as a result, the task generates a sequence of jobs during its execution. In Fig. 1, activated job sequences for each task are shown by downward arrow. Furthermore, for showing the k'th instance (job) of a task τ_i, we use notation $J(i, k)$. In this example, the first two tasks (τ_1, τ_2) are HCTs (from level A in Table 1) while others are LCTs. LCTs consist of both MCTs, which are from the levels $\{B, C\}$ in Table 1 and non-MCTS from level $\{D, E\}$. In addition, we assume that MCTs should not be dropped frequently in the HI mode due to the occurrence of catastrophic consequence (which we discussed in previous sections). Hence, in this example, we focus on our scheduling policy and we can suppose that

TABLE 4. Example of MC task set.

	ζ_i	C_i^{LO}	C_i^{HI}	T_i	$\hat{d}_i$	δ_i
τ_1	*HI*	1	5	12	6	∞
τ_2	*HI*	1	2	24	12	∞
τ_3	*LO*	1	1	4	-	3
τ_4	*LO*	1	1	3	-	4
τ_5	*LO*	1	1	6	-	1

*$\tau_1, \tau_2 \in \{A\}$, $\tau_3, \tau_4 \in \{B, C\}$ and $\tau_5 \in \{D, E\}$

the measurement of the execution time will be accomplished after applying the fault-tolerance technique [8]. In addition, in this example, all of the tasks should comply with their specified time budget (C_i^{LO} in the LO mode and C_i^{HI} in the HI mode) to be executed correctly. Our task set is able to be scheduled under EDF-VD [7] (for more information, the reader is referred to Section III-F). Also, the virtual deadlines ($\hat{d}_i$) of the HCTs are computed and mentioned in Table 4 (virtual deadlines are less than the actual deadlines and provide a higher priority for HCTs). For simplicity of this example, we round the virtual deadlines into acceptable integers. Suppose that, when the first job of the τ_1 (which is denoted as J(1,1)) is being executed, due to the occurrence of a fault, the system switches to the HI mode. As shown in Fig. 1, the system switches at timeslot 3 in this example.

The operation of the EDF-VD scheduling algorithm (by considering task killing) to the task set under [14] and [8]

has been shown in Fig. 1(a). Whenever the system switches to the HI mode, the active jobs of LCTs will be dropped until the system would safely switch back to the LO mode (at time 9 in Fig. 1(a) when there is no active HCT). According to Fig. 1(a), the active jobs of LCTs, which are MCT (τ_4) (J(4,2) and J(4,3)) are dropped twice, which is not tolerable for these tasks (according to the Table 4, this task can be dropped once in each $\delta_4 \times T_4$ in the HI mode).

On the other hand, consider a situation that MCTs (τ_3, τ_4) would not be dropped in the HI mode. Actually, the execution of these tasks becomes as important as the execution of HCTs. Therefore, τ_5 as an LCT is the only task, which is dropped in the HI mode. Similar to the previous example, EDF-VD algorithm is applied to the task set. Based on Fig. 1b, when the system switches to the HI mode, according to the EDF-VD policy, the first job of HCT τ_2 (J(2,1)) is not executed in its predefined period and it will miss its deadline at time 24. Such scenarios of task scheduling may cause catastrophic consequences due to deadline missing of HCTs.

Now, consider a parameter δ_i for LCTs. When the system switches to the HI mode due to the execution time of HCTs, LCTs will be dropped in every δ_i to create slack time for the HCTs to be executed before their deadlines. According to Fig. 1c, when the system switches to the HI mode, the first job of the MCTs τ_3, and τ_4 are dropped, which are J(3,2), J(4,2). Consequently, all of the HCTs will be executed in this HI mode. Hence, since $\delta_5 = 1$, it would be always dropped in the HI mode. At the same time, by employing this technique, the MCTs would not be frequently dropped, which is desirable.

V. PROPOSED METHOD: FANTOM

In this section, we briefly introduce the quantification of MC tasks in Section V-A. Then, in Section V-B, we define MC task utilizations and based on them, we present the scheduling analysis technique for the proposed MC task model (FANTOM) in Section V-C and V-D. In the end, a general design-time scheduling algorithm is presented in Section V-E, in which all essential conditions that must be guaranteed are determined.

A. SAFETY QUANTIFICATION

Independent from the level of criticality, any of the jobs in the tasks are executed up to n_i times to guarantee their safety requirements with regards to PFHs, which is presented in Table 1 and Section III-D [8], [11]. Here, n_{HCT} and n_{LCT} are the maximum required re-execution times based on the PFH of both high- and low-criticality levels. However, in the worst-case, if jobs execute n_i times to satisfy the safety requirements, it may cause the system to be overloaded (i.e., $U_{sys} > 1$) and lead the system to be unschedulable. Further, another time constraint n'_{HCT} ($n'_{HCT} < n_{HCT}$) has been defined for HCTs, that causes the system to operate without being overloaded. In addition, this time constraint gives the system this ability to switch to the HI mode when the correct response is not ready (e.g., due to a fault occurrence) after executing it for n'_{HCT} times. In this case, we use our proposed drop-aware policy for LCTs to guarantee the safe execution of HCTs. Hence, n'_{HCT} is the highest possible value that causes the system to be schedulable in the LO mode. Now, the WCET in each criticality level is computed as follows:

- LCTs: $C_i^{LO} = C_i^{HI} = n_{LCT} \times C_i$
- HCTs: $\begin{cases} C_i^{HI} = n_{HCT} \times C_i \\ C_i^{LO} = n'_{HCT} \times C_i \end{cases}$

According to [8], the values of n_{HCT} and n_{LCT} for tasks in the same level of criticality is computed by solving the Eq. (2), which has exploited the PFH (probability-of-failure-per-hour, which we previously defined) of LCTs and HCTs based on Table 1. Hence, for each level, the value of PFH is the same from one hour to the next hour. Also, in Eq. (2), the HP is the hyper period of all tasks. Since the unit of the $pfh(\zeta)$ is hour, the HP$'$ represents the hyper period, in the unit of hour [8]. In this equation, $max(\lfloor \frac{HP - n_i \times C_i}{T_i} + 1 \rfloor, 0)$ represents the maximum number of execution rounds for task τ_i in the hyper period $(0, HP]$. Moreover, as we presented in section III-C, f_i is a probability factor, which indicates the probability of an unsuccessful execution for a task due to transient faults (i.e., PoF). So, $f_i^{n_i}$ represents that a task is executed n_i times in the worst case to have successful execution but it fails in all executions. Therefore, the failure probability per hour for each criticality level ($pfh(\zeta)$) can be calculated by Eq. (2) [8]. As can be realized from this equation, by increasing the value of n_ζ (ζ = HCT or LCT), $pfh(\zeta)$ is decreased. Therefore, the minimum value of n_ζ for each criticality level is computed when $pfh(\zeta) \leq PFH_\zeta$.

$$pfh(\zeta) = \frac{(\sum_{\tau_i \in \tau_\zeta} max(\lfloor \frac{HP - n_\zeta \times C_i}{T_i} + 1 \rfloor, 0) \times f_i^{n_i})}{HP'} \quad (2)$$

In the next step, as we discussed, the value of n'_{HCT} has to be computed in a way that the system would be schedulable and all the safety requirements are met. As we mentioned before, all LCTs should be executed correctly before their deadline in the LO mode. Hence, by assigning n_{LCT} to LCTs, the number of re-executions for HCTs in the LO mode (n'_{HCT}) to have the schedulable system will be computed by solving Eq. (3) that $pfh(LO) < PFH_{LO}$ [8].

$$pfh(LO) = \frac{(1 - \prod_{\tau_i \in \tau_{HCT}} (1 - f_i^{n'_i})^{max(\lfloor \frac{HP - n'_i \times C_i}{T_i} + 1 \rfloor, 0)}) \times w(\infty, HP)}{HP'} \quad (3)$$

In this equation, the maximum number of execution rounds for HCT τ_i in each hyper period ($[0, HP]$), that in each round, it is executed n'_{HCT} times, is $max(\lfloor \frac{HP - n'_{HCT} \times C_i}{T_i} + 1 \rfloor, 0)$. Since in the LO mode, HCTs are not executed more than n'_{HCT} times to guarantee task schedulability, the probability that no job of HCTs executes more than n'_{HCT} times in each hyper period is bounded by $P = \prod_{\tau_i \in \tau_{HCT}} (1 - f_i^{n'_{HCT}})^{max(\lfloor \frac{HP - n'_{HCT} \times C_i}{T_i} + 1 \rfloor, 0)})$ that all HCTs are executed successfully in maximum n'_{HCT}

times and no LCT is dropped. Therefore, the probability that LCTs are dropped is $1 - P$. Due to the characteristics of our system, the maximum PFH for LCTs, $w(\infty, HP)$, is defined and computed differently from what was defined in [8]. To compute this function, we should obtain the maximum number of executions for each task that can be done in one hyper period. Normally, this number is accommodated by $max\{\left\lfloor \frac{HP-n_{LCT}\times C_i}{T_i} + 1 \right\rfloor, 0\}$. Due to the newly defined parameter (δ_i) for the tasks, this round number is obtained and used in function *w(HP)* as:

$$w(\delta, HP) = \sum_{\tau_i \in \tau_{LCT}} max((\lfloor \frac{HP - n_{LCT} \times C_i}{T_i} + 1 \rfloor - \lfloor \frac{HP - n_{LCT} \times C_i}{T_i \times \delta_i} + 1 \rfloor), 0) \times f_i^{n_{LCT}} \quad (4)$$

To find the maximum *pfh* for LCTs in Eq. (3) that $pfh(LO) < PFH_{LO}$, parameter δ_i for each LCT in Eq. (4) should be infinitive. It means, no LCT would be dropped in the LO mode. Therefore, as can be seen in Eq. (3), the value of n'_{HCT} is independent of the values of δ_i of LCTs that is used in Eq. (4). It should be noted that all HCTs have the same value of n'_{HCT} for their execution in the LO mode.

B. MC TASKS UTILIZATION BOUNDS DEFINITION

In this section, we present the different utilization bounds for MC tasks, which are used in the task scheduling. Based on the description of safety requirements presented in Section V-A, the utilization of task j at level k is defined as $u_j^k = (C_j^k)/T_j$, in which, if task j is an LCT, $C_j^k = n_{LCT} \times C_j$ and if task j is an HCT, $C_j^k = n_{HCT} \times C_j$ with k: LO, and $C_j^k = n'_{HCT} \times C_j$ with k: HI. According to this definition, the low and high bound of utilization for different modes of task τ_j will be represented as u_j^{LO} and u_j^{HI}, respectively. Thus, the low-level and high-level utilization of HCTs and also the low bound utilization of LCTs are defined as follows:

$$\begin{cases} U_{HCT}^{LO} = \sum_{\zeta_j=HCTs} u_j^{LO} \\ U_{HCT}^{HI} = \sum_{\zeta_j=HCTs} u_j^{HI} \end{cases} \quad (5)$$

$$U_{LCT}^{LO} = \sum_{\zeta_j=LCTs} u_j^{LO} \quad (6)$$

Theorem 1: Due to the execution of some LCTs, in the HI mode, the high bound utilization is presented as follows:

$$U_{LCT}^{HI} = \sum_{\zeta_j=LCTs} u_j^{LO} \times \frac{(\delta_j - 1)}{\delta_j} = u_j^{HI} \quad (7)$$

Proof: Since we have to guarantee the correct execution of all HCTs in the HI mode, few LCTs will be dropped in this mode. Therefore, we need to consider the jobs of LCTs that are released in this HI mode. Due to the intended feature of LCTs, one job could be dropped in every δ_j job instances in the HI mode. Accordingly, for these tasks, $\delta_j - 1$ jobs must be executed among δ_j jobs (i.e., if we have a period of time $\delta_j \times T_j$, LCTs are executed for a time equal to $C_j^{HI} \times (\delta_j - 1)$ in this period, $u_j^{HI} = \frac{C_j^{HI} \times (\delta_j - 1)}{\delta_j \times T_j}$). Also, we have assumed that for each LCT, the value of C_j^{LO} is equal to C_j^{HI}. Therefore, the high bound utilization is rewritten as follows, which is lower than U_{LCT}^{LO} (for each LCT, $u_j^{HI} = u_j^{LO} \times \frac{(\delta_j - 1)}{\delta_j} < u_j^{LO}$):

$$U_{LCT}^{HI} = \sum_{\zeta_j=LCTs} (\frac{C_j^{HI} \times (\delta_j - 1)}{T_j \times \delta_j} = u_j^{LO} \times \frac{(\delta_j - 1)}{\delta_j} = u_j^{HI}) \quad (8)$$

■

Hence, in equality (7), if $\delta_j = 1$ for an LCT, then the utilization of this task j in the HI mode (u_j^{HI}) would be equal to 0. In other words, these tasks are not executed in the HI mode and then, FANTOM uses task dropping for these LCTs with $\delta_j = 1$ (such as non-MC tasks) in the HI mode due to our proposed policy.

C. SCHEDULING ANALYSIS

To guarantee the correct execution of MC tasks before their deadlines, several conditions have to be met at design time. We investigate the MC tasks schedulability in different system behaviour as:

- Guaranteeing the task schedulability in the LO mode.
- Guaranteeing the task schedulability in case of mode switching and then, in the HI mode.
- Guaranteeing the task schedulability while the EDF-VD scheduling algorithm is used.

The conditions for each item are explained in detail in this section. In the end, the last condition based on the system utilization is presented in Section V-D. As mentioned before, at run-time, the MC system initially operates in the LO mode under EDF-VD. In FANTOM, when the system switches to the HI mode, with respect to the execution of HCTs, the first job of LCTs will be dropped. These LCTs will be dropped periodically in an interval equal to $\delta_j \times T_j$ until the system is in the HI mode. In the meantime, all the HCTs will be executed. Hence, if the mission of an LCT is more important that other LCTs, the value of δ for this LCT would be higher. The details are as follows.

1) CONDITIONS TO GUARANTEE TASK SCHEDULABILITY IN THE LO MODE

By using the notations and definitions, we can easily express that MC task sets are schedulable under EDF if the following condition is guaranteed in a core in the LO mode.

$$U_{HCT}^{LO} + U_{LCT}^{LO} \leq 1 \quad (9)$$

As we have mentioned, due to using the EDF-VD algorithm, deadlines of HCTs will be downscaled by a multiplication factor x in the LO mode. Hence, the low bound utilization of HCTs (U_{HCT}^{LO}) would be downscaled by *1/x*. Based on the EDF-VD algorithm [14], due to the executing of LCTs in the HI mode by using the parameter δ, we provide a problem formulation.

The following condition (which is obtained by the modification of the inequality (9)) is sufficient to schedule all

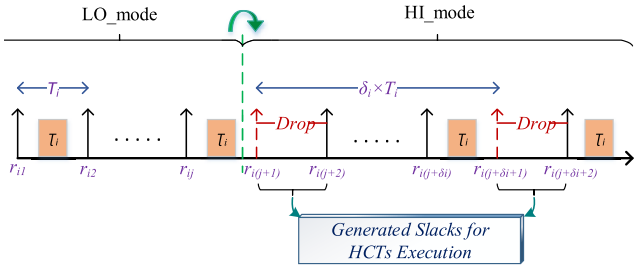

FIGURE 2. LCTs scheduling solution in both modes by dropping one job every δ_i in task τ_i in the HI mode .

of the tasks by the EDF-VD algorithm in the LO mode [7]. As mentioned, $U^{LO}_{HCT} = \sum_{\zeta_i=HCT} \frac{n_{HCT} \times C_i}{T_i}$. Since $d_i = T_i$, and the virtual deadline $\hat{d}_i = x \times d_i$ is used for HCTs in the LO mode to schedule tasks, therefore the utilization of HCTs in the LO mode is $\frac{U^{LO}_{HCT}}{x}$ that is used for task schedulability test.

$$\frac{U^{LO}_{HCT}}{x} + U^{LO}_{LCT} \leq 1 \tag{10}$$

2) CONDITIONS TO GUARANTEE TASK SCHEDULABILITY IN THE HI MODE

Since the MC systems must work successfully in the HI mode, we introduce a theorem and conditions to guarantee the deadline meeting of both HCTs and LCTs in the HI mode. Before introducing a new theorem, we explain how the new parameter δ is used. As depicted in Fig. 2, when the system switches to the HI mode, the demand requested for core computation time by the tasks is increased. In such critical situations, the first job of the LCTs is dropped and consequently, one job of each τ_i is dropped every δ_i until the system switches back to the LO mode. These generated slacks are used to execute HCTs in the HI mode. With this value of δ for each task, we determine a theorem for ensuring that both HCTs and LCTs (with $\delta > 1$) are scheduled within their deadlines, in the HI mode.

Theorem 2: The sufficient establishing condition for executing both HCTs and LCTs in the HI mode is presented in inequality (11), in which, in the worst case, the system remains in the HI mode for the whole hyper period. In this inequality, the HP is the hyper period of HCTs and LCTs with $\delta > 1$ in a processing unit.

$$\frac{\sum_{\zeta_j \in HCT} \lfloor \frac{HP}{T_j} \rfloor \times C^{HI}_j}{HP} + \frac{\sum_{\zeta_j \in LCT, \delta_j > 1} (\lfloor \frac{HP}{T_j} \rfloor - \lfloor \frac{HP}{T_j \times \delta_j} \rfloor) \times C^{HI}_j}{HP} \leq 1 \tag{11}$$

Proof: Due to the execution of HCTs and LCTs in the HI mode, assume that the first task, which starts its execution is an HCT and this task causes the system to switch to the HI mode. In the worst case, the system remains in the HI mode for the whole hyper period. In this HI mode, the maximum time interval in which HCTs are executed in one HP ($TInterval^{max}_{HCT}$) is as:

$$TInterval^{max}_{HCT} = \sum_{\zeta_j \in HCT} \lfloor \frac{HP}{T_j} \rfloor \times C^{HI}_j \tag{12}$$

In addition, due to the execution of LCTs in the HI mode and the nature of these tasks, this maximum time interval that LCTs can be executed in one HP ($TInterval^{max}_{LCT}$) is as:

$$TInterval^{max}_{LCT} = \sum_{\zeta_j \in LCT, \delta_j > 1} (\lfloor \frac{HP}{T_j} \rfloor - \lfloor \frac{HP}{T_j \times \delta_j} \rfloor) \times C^{HI}_j \tag{13}$$

Accordingly, if these two types of tasks need to be schedulable by the EDF algorithm in the HI mode before their deadlines, the following inequality must be guaranteed ($TInterval^{max}_{HCT} + TInterval^{max}_{LCT} \leq HP$).

$$\sum_{\zeta_j \in HCT} \lfloor \frac{HP}{T_j} \rfloor \times C^{HI}_j + \sum_{\zeta_j \in LCT \& \delta_j > 1} (\lfloor \frac{HP}{T_j} \rfloor - \lfloor \frac{HP}{T_j \times \delta_j} \rfloor) \times C^{HI}_j \leq HP \tag{14}$$

By dividing the both sides of this inequality by HP, the inequality (11) will be obtained. Inequality (11) is the establishing condition for the schedulability of the tasks. ■

According to the EDF-VD algorithm, there are some scenarios in which the inequality (10) has been satisfied, while HCTs in the HI mode have missed their deadline. Hence, a condition should be considered and satisfied in order to guarantee the schedulability of all of the HCTs and LCTs (based on parameter δ) in their specified deadlines by the EDF-VD in the HI mode. Besides, there is a scenario that a task is released before mode switching while its deadline is after mode switching and does not finish its execution yet, called *carry-over job* [4], [20], [21]. To consider the carry-over problem, the following sufficient condition has been expressed (To know about the proof of this condition, the reader is referred to Appendix A, which is the same lemma and has the same proving flow presented in [20], [28]).

$$U^{HI}_{HCT} + (1 - x) \times U^{HI}_{LCT} + x \times (U^{LO}_{LCT}) \leq 1 \tag{15}$$

While the inequalities (15) and (11) are not necessary (just sufficient), the necessary condition would be driven when the sum of the utilization of LCTs (i.e., MCTs) and HCTs in the HI mode are higher than 1 or to guarantee the correct execution of jobs of each task before their individual deadlines in this HI mode. Thus, the necessary condition for scheduling both HCTs and LCTs by the EDF algorithm in the HI mode and being executed correctly before their deadlines, is the following condition.

$$U^{HI}_{HCT} + U^{HI}_{LCT} \leq 1 \tag{16}$$

3) CONDITIONS TO GUARANTEE TASK SCHEDULABILITY WITH EDF-VD ALGORITHM

Now, we present the value of x to obtain the virtual deadline by multiplying the actual deadline by x. Then, we present a new condition based on the previous conditions and the

EDF-VD algorithm. By considering the inequalities (10) and (15), it could be concluded that the value of $x\,(d'_j = x \times d_j)$ is obtained through the inequality (17).

$$\frac{U^{LO}_{HCT}}{1-(U^{LO}_{LCT})} \leq x \leq \frac{1-(U^{HI}_{HCT}+U^{HI}_{LCT})}{(U^{LO}_{LCT})-U^{HI}_{LCT}} \tag{17}$$

Based on the interval for x in this inequality, and according to expression (18), the EDF-VD algorithm chooses the smallest value for x [7]. In addition, as explained before, we use the fault tolerance technique, re-execution, to guarantee the correct execution of all tasks within their safety requirements based on Table 1 in any circumstance. To show how the parameters such as safety requirements and virtual deadlines, affect each other, we can rephrase the value of x as follows. In this expression, when a task is executed and a fault occurs, it needs to to be re-executed for a maximum of $(n_{\zeta_j}-1)$ times to guarantee its safety requirement that ζ_j is LCT or HCT. Hence, the value of x is independent from δ_j. As mentioned in Section V-A, the values of parameter δ_j has no effect on computing n'_{HCT} (based on Eq. (3), (4)).

$$x \leftarrow \frac{U^{LO}_{HCT}}{1-(U^{LO}_{LCT})} \implies x \leftarrow \frac{n'_{HCT} \times \sum_{j\in HCT} C_j/T_j}{1-(n_{LCT} \times \sum_{j\in LCT} C_j/T_j)} \tag{18}$$

In addition to the mentioned conditions, another condition is required to guarantee that the task set would be schedulable by EDF-VD. In this regard, the upper bound utilization of the system will be computed by exploiting the condition (17), and represented as:

$$\begin{aligned} \frac{U^{LO}_{HCT}}{1-U^{LO}_{LCT}} &\leq \frac{1-(U^{HI}_{HCT}+U^{HI}_{LCT})}{U^{LO}_{LCT}-U^{HI}_{LCT}} \\ &\iff U^{LO}_{HCT} \times (U^{LO}_{LCT}-U^{HI}_{LCT}) \\ &\leq (1-U^{HI}_{HCT}-U^{HI}_{LCT}) \\ &\quad \times(1-U^{LO}_{LCT}) \iff U^{HI}_{HCT} \\ &\leq 1-U^{HI}_{LCT}-\frac{U^{LO}_{HCT}\times(U^{LO}_{LCT}-U^{HI}_{LCT})}{1-U^{LO}_{LCT}} \end{aligned} \tag{19}$$

According to inequality (19), the upper bound utilization of HCTs in the HI mode (U^{HI}_{HCT}) is obtained. If this condition is satisfied, the given task set is schedulable by the EDF-VD algorithm under the conditions in FANTOM.

Generally, inspired by the presented conditions, the condition that should be investigated in a core to guarantee that a task set is schedulable under EDF-VD algorithm in FANTOM are inequalities (11) and (20). Condition (20) is obtained by using inequalities (9) and (19).

$$\begin{aligned} max(U^{LO}_{HCT}+U^{LO}_{LCT}, U^{HI}_{HCT}+U^{HI}_{LCT} \\ +\frac{U^{LO}_{HCT}\times(U^{LO}_{LCT}-U^{HI}_{LCT})}{1-U^{LO}_{LCT}}) \leq 1 \end{aligned} \tag{20}$$

D. SYSTEM UPPER BOUND UTILIZATION

In this section, we present the system upper bound utilization in order to enable MC tasks to be schedulable by the FANTOM. In the end, we present the last condition that must be guaranteed. We nominate U_p as an upper bound for the task set, which should be schedulable in both HI and LO modes. This bound is defined as follows:

$$U_p = max(U^{LO}_{HCT}+U^{LO}_{LCT}, U^{HI}_{HCT}+U^{HI}_{LCT}) \tag{21}$$

We have explained that the condition (17) is sufficient for the task sets to be schedulable by the EDF-VD algorithm. Hence, by using conditions (17) and (21), the following expression could be derived. Here, our goal is to find the U_p, which still satisfies the following expression. Thereby, we have:

$$\frac{U^{LO}_{HCT}}{1-U^{LO}_{LCT}} \leq \frac{1-(U^{HI}_{HCT}+U^{HI}_{LCT})}{U^{LO}_{LCT}-U^{HI}_{LCT}} \tag{22}$$

Since we have $U^{LO}_{HCT}+U^{LO}_{LCT} \leq U_p \Rightarrow (U^{LO}_{HCT} \leq U_p - U^{LO}_{LCT}$ and also $U^{HI}_{HCT}+U^{HI}_{LCT} \leq U_p \Rightarrow 1-(U^{HI}_{HCT}+U^{HI}_{LCT}) \leq 1-U_p$:

$$\frac{U_p-(U^{LO}_{LCT})}{1-(U^{LO}_{LCT})} \leq \frac{1-U_p}{(U^{LO}_{LCT})-U^{HI}_{LCT}} \tag{23}$$

This condition will be satisfied if and only if:

$$\begin{aligned} (U^{LO}_{LCT})^2 - U^{LO}_{LCT} \times (1+U^{HI}_{LCT}) + 1 + U_p \times (-1 \\ +U^{HI}_{LCT}) \geq 0 \end{aligned} \tag{24}$$

Accordingly, if Eq. (25) is met, expression (26) will be obtained according to the expression (24) (which is always true for each of the low-criticality utilization in [0, 1)).

$$1+U_p \times (-1+U^{HI}_{LCT}) = \frac{(1+U^{HI}_{LCT})^2}{4} \tag{25}$$

$$(U^{LO}_{LCT}-\frac{1+U^{HI}_{LCT}}{2})^2 \geq 0 \tag{26}$$

By simplification of Eq. (25), it will turn into:

$$U_p = \frac{3+U^{HI}_{LCT}}{4} \tag{27}$$

Accordingly, it could be concluded that the upper bound (U_p) depends on the utilization of the LCTs in the HI mode. It means, (U_p) depends on the parameter of δ_j, which is different for each LCT $(U^{HI}_{LCT} = \sum_{j\in LCTs}((\delta_j-1)\times C_j)/(\delta_j \times T_j))$. Since the U_p is the utilization bound of the system, which is run on a single-core processor, the maximum value of it is 1. In the case of $U^{LO}_{LCT}+U^{LO}_{HCT} < U^{HI}_{LCT}+U^{HI}_{HCT}$, then, $U_p = U^{HI}_{LCT}+U^{HI}_{HCT}$. Therefore, according to equality (27), in addition to inequality (19), another condition and upper bound for U^{HI}_{HCT} should be checked to guarantee the schedulability of a task set in the HI mode, if we have $U^{LO}_{LCT}+U^{LO}_{HCT} < U^{HI}_{LCT}+U^{HI}_{HCT}$, which is:

$$U^{HI}_{HCT} \leq \frac{3(1-U^{HI}_{LCT})}{4} \tag{28}$$

E. A GENERAL DESIGN TIME SCHEDULING ALGORITHM

Now, we review the proposed approach algorithm at design-time and show which of the presented conditions need to be checked. The pseudo code of our scheduling algorithm has been illustrated in Algorithm 1, which explains the mechanism of the schedulability test. In summary, at the beginning and according to Eq. (2), we calculate the re-execution profiles for each task (either high or low). Also, we calculate the minimum re-execution profiles for HCTs through Eq. (3) (line 1). Subsequently, the utilizations are calculated (line 2). In addition, we calculate the maximum re-execution profiles for HCTs through schedulability test (line 3). If the maximum re-execution profile is more than the minimum one, we select this amount as n'_{HCT} and consequently, the utilization of HCTs in the LO mode is calculated. Otherwise, the algorithm will return a false value (lines 4-8). According to Algorithm 1, at the first stage, FANTOM evaluates the utilization bound in both LO and HI modes. If they are less than 1, it means the task set can be scheduled by the EDF in both modes (lines 10-13). Otherwise, the inequality (11) is evaluated in order to check whether both of the HCTs and LCTs with $\delta_j > 1$ (which are MCT), are executed in the HI mode or not. In addition, the two mentioned conditions in inequality (20) and also Eq. (28) in case of $U^{LO}_{LCT} + U^{LO}_{HCT} < U^{HI}_{LCT} + U^{HI}_{HCT}$, will be evaluated to test the schedulability of the task set (line 15). If all the conditions are met, the virtual deadline coefficient will be assigned with the minimum value (line 16). Hence, the virtual deadlines of HCTs will be obtained by using the mentioned value in Eq. (18). Then, the sufficient condition of Eq. (15) is checked (line 17). If this equation is satisfied and the algorithm returns the true value, the task set will be scheduled by the EDF-VD algorithm (in which, all LCTs will be dropped in every δ_j in the HI mode (lines 18-20)). On the other hand, if none of the conditions are met, the task set cannot be scheduled and the algorithm will return a false value (lines 22, 25).

Algorithm 1 Design-Time Scheduling Method Pseudo Code

Schedulability Test(Task Set)

1: (n_{LCT}, n_{HCT}) are obtained by Eq. (2) & (n'_{HCT}) by Eq. (3)
2: $(U_{HCT}, U^{HI}_{HCT}, U^{LO}_{LCT}, U^{HI}_{LCT}) \leftarrow$ *Util_Computation (taskset,*n_{LCT}, n_{HCT} *)*
3: $n'_2 = \sup\{max(n \times U_{HCT} + U^{LO}_{LCT}, U^{HI}_{HCT} + U^{HI}_{LCT} + \frac{n \times U_{HI} \times (U^{LO}_{LCT} - U^{HI}_{LCT})}{1 - U^{LO}_{LCT}}) \leq 1\}$;
4: **if** $n'_{HCT} < n'_2$ **then**
5: $\quad n'_{HCT} = n'_2$
6: **else**
7: $\quad$ **return** "The task set is not schedulable"
8: **end if**
9: $(U^{LO}_{HCT}) \leftarrow$ *Util_Computation (taskset,*n'_{HCT} *)*
10: **if** $U^{HI}_{HCT} + U^{HI}_{LCT} \leq 1$ & $U^{LO}_{HCT} + U^{LO}_{LCT} \leq 1$ **then**
11: $\quad \hat{T}_i \leftarrow T_i$ for all tasks
12: $\quad$ Schedule Task set with EDF Algorithm
13: $\quad$ **return** "The task set is schedulable"
14: **else**
15: $\quad$ **if** *Eq.* (11) & *Eq.* (20) are satisfied & $[(U^{LO}_{LCT} + U^{LO}_{HCT} < U^{HI}_{LCT} + U^{HI}_{HCT}$ & *Eq.* (28)) **or** $(U^{LO}_{LCT} + U^{LO}_{HCT} \geq U^{HI}_{LCT} + U^{HI}_{HCT})]$ **then**
16: $\quad\quad x \leftarrow$ is computed by Eq. (18)
17: $\quad\quad$ **if** Eq. (15) is satisfied **then**
18: $\quad\quad\quad \hat{T}_i \leftarrow T_i \times x$ for each high-criticality task
19: $\quad\quad\quad$ Schedule Tasks with modified EDF-VD
20: $\quad\quad\quad$ **return** "The task set is schedulable"
21: $\quad\quad$ **else**
22: $\quad\quad\quad$ **return** "The task set is not schedulable"
23: $\quad\quad$ **end if**
24: $\quad$ **else**
25: $\quad\quad$ **return** "The task set is not schedulable"
26: $\quad$ **end if**
27: **end if**

VI. EXPERIMENTAL RESULTS

In this section, the experimental results of the FANTOM are validated through extensive simulations on two case studies from avionics domain presented in [8] and [49]. Then, we evaluate the impact of MC task's parameter variations in the schedulability test of the task sets.

A. FLIGHT MANAGEMENT SYSTEM

1) FIRST CASE STUDY

Avionic Real-Life applications have been used in different papers to evaluate their presented methods [3], [8], [12]. To evaluate our method, we use the FMS application introduced in [8], which consists of 7 tasks from level B and 4 tasks from level C (Table 1). We consider the tasks from level B as HCTs and the tasks from level C as the MCTs. Therefore, there are no non-critical tasks in this task set. We can also define different values for the PoF of each task. To evaluate this part, this parameter is assumed to be 10^{-5} [8], [11]. In addition, the value of the skip parameter for all LCTs is considered to be $\delta = 4$. According to Eq. (2) and (3), the number of re-executions for all of the tasks is calculated and set to $n_{LCT} = n_{HCT} = 3$ and $n'_{HCT} = 2$, respectively, to guarantee the safety requirements of the tasks without task killing and service degradation. Fig. 3 represents the impact of FANTOM on the system schedulability. As shown in this figure, by increasing n'_{HCT}, the utilization of the system will be increased due to the HCTs low utilization increment in inequality (20). In addition, the system will no longer be schedulable when $n'_{HCT} > 2$. On the other hand, by increasing n'_{HCT} (the redundancy for HCTs needs to be increased), the PFH of LCTs will be decreased and consequently, the system's safety could be improved. In essence, the probability of mode switching would be decreased and as a result, the LCTs will be dropped less likely. Hence, by assigning $n'_{HCT} = 2$, PFH of LCTs will be 10^{-10} and the utilization will be 0.95, which is less than 1.

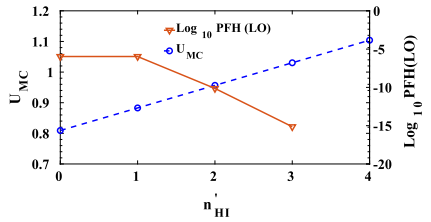

FIGURE 3. FANTOM implementation for FMS application: case study 1.

2) SECOND CASE STUDY

Here, we investigate a set of real tasks, in which, HCTs have the safety requirement of level A. There is a case study for FMS application, introduced in [49], which consists of four tasks, three tasks from level A (responsible for executing the necessary control steps and essential for a reliable flight behaviour), and one task from level B (responsible for detecting the objects). For this example, the number of re-executions is set to $n_{HCT} = 3$, $n_{LCT} = 1$ and $n'_{HCT} = 2$, respectively. By having the same setting as the previous case study, Fig. 4 depicts the impact of FANTOM on system schedulability and LCT's PFH. As shown, the system is not schedulable for $n'_{HI} > 2$. Therefore, by assigning $n'_{HI} = 2$, PFH of LCT is 10^{-9} ($<10^{-7}$) and $U_{MC} = 0.96$.

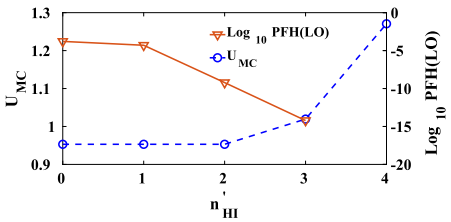

FIGURE 4. FANTOM implementation for FMS application: case study 2.

In general, the safety requirement of HCTs/LCTs affects the number of re-executions (n_{HCT}/ n_{LCT}), and consequently, the utilization and the number of re-executions for HCTs in the LO mode (according to Eq. (3)) are changed. Hence, according to the Eq. (2), the number of re-executions for HCTs and LCTs depends on the task properties, such as WCET, PFH and the number of tasks in each level. For example, for the same number of tasks and WCET for each task, if PFH is changed from 10^{-7} to 10^{-9} for the tasks' level, the number of re-execution may be increased to guarantee the safety requirement.

B. EXPERIMENTAL SETUP AND EXTENSIVE RESULTS

In the following, the FANTOM has been evaluated by exploiting random MC task sets. These sets have been generated through the provided technique in [7], [8]. As an input parameter, the system's utilization (U_{bound}) is obtained as equality (21), which should be less than 1. At the beginning, the U_{bound} for the generated tasks are set to zero (i.e., the task system is initialized to be empty) and afterward, new tasks will be added to the task set in a random manner to increase the U_{bound} until a certain value (U_{bound} is increased with steps of 0.05). The period (T) and utilization of the tasks are generated uniformly within the range of [10, 100] and [0.01, 0.1], respectively. According to conditions, for each data-point (i.e., U_{bound}) in the range of [0.05, 1], 1000 task sets are generated and evaluated from the schedulability and fault occurrence perspectives. In the end, the ratio of task sets, which were deemed as schedulable, will be reported.

In the established simulations, we have considered HCTs from level A, LCTs from level B to E in which, MCTs from level B or C, and non MCTs from level D and E. Furthermore, the value of the parameter (δ) is randomly generated between one and the maximum amount of this parameter for LCTs. As the maximum amount of this parameter is considered to be determined by the designer, we investigate the results by varying the maximum amount in the range of [2,16] in the next subsection (Section VI-B1).

The efficiency of FANTOM has been investigated through extensive simulations and its comparison with the provided algorithms in [8] and [15]. As we mentioned before, researchers in [8] use EDF-VD to schedule the tasks. Also, researchers in [15] use the FP scheduling algorithm that is based on applying Response Time Analysis (RTA). In this regard, our observations are categorized into five subsections. There are some graphs relating to the results in which the y-axis represents the fraction of schedulable task sets, which is called acceptance ratio, and the x-axis represents utilization. It should be noted that since the *Ubound* shows the utilization of task sets before applying the fault tolerance technique and calculating the utilization in the LO and HI modes, the maximum utilization bound that the task set is schedulable has a small amount in graphs.

1) EFFECT OF VARYING LCT's PARAMETER (δ)

At the beginning, we evaluate the effects of varying maximum value of the newly defined parameter (δ) for LCTs. As a result, according to Fig. 5, by reducing the maximum value of parameter δ, the acceptance ratio will be slightly improved. The reason is that, in the case of increasing this parameter, the utilization of LCTs in the HI mode will be increased due to Eq. (7). Therefore, the system schedulability will be decreased considerably according to inequality (20). If $max(\delta) = 1$, it means all LCTs are non-critical and dropped in the HI mode. Indeed, there is no MCT in the system. In this case, the acceptance ratio of the proposed approach is the

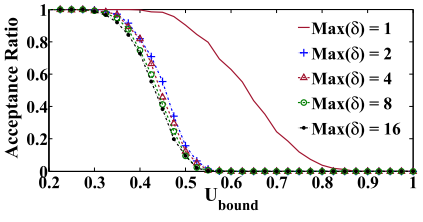

FIGURE 5. Acceptance ratio with varying the parameter (δ) for LCTs.

same as the acceptance ratio of the method, proposed in [8], in which all LCTs are dropped when the system switches to the HI mode. Indeed, there is no restriction for the system safety when none of LCTs are relevant to it. Hence, the significance of the proposed method is when there are some MCTs in the system and in this case, the proposed method performs better than [8].

In the rest of this article, we consider $max(\delta) = 4$ to show the efficiency of our proposed method.

2) ACCEPTANCE RATIO OF SCHEDULABLE TASK SETS

We further compare the fraction of task sets, which could be scheduled under FANTOM, the traditional fault tolerant EDF-VD algorithm [8], and fault-tolerant FP algorithm [15]. This fraction is defined as the acceptance ratio. Researchers in [8] have considered both task killing and service degradation for LCTs in the HI mode. We use task killing of LCTs by considering the QoS of LCTs and execute them as much as possible in the HI mode to have a fair comparison. In addition, researchers in [15] have striven to increase the QoS of LCTs in the HI mode, which we examine here. Accordingly, in this subsection, we have assumed that 40% of them are HCTs and 60% are LCTs that 30% of LCTs are considered as MCTs and the remaining 30% are non-MCTs (with $\delta = 1$). In should be noted, a task set is schedulable if all tasks can be scheduled in the LO mode, and then after switching to the HI mode, all HCTs would be executed and also, LCTs cannot be frequently dropped. As shown in Fig. 6, when the utilization of the system is smaller than 0.225, the tasks are always schedulable by both algorithms, which use EDF-VD. By increasing the utilization bound, the proposed algorithm could always schedule the task sets as long as the utilization is smaller than 0.275. Furthermore, Fig. 6 explains that the proposed algorithm can improve the acceptance ratio by up to 43.9% and 65.9% compared to the traditional fault tolerant EDF-VD algorithm [8] and fault tolerant FP algorithm [15], respectively. Since the EDF-VD is used in both our proposed method and [8], in the rest of paper, we show the effectiveness of our proposed method in comparison with [8].

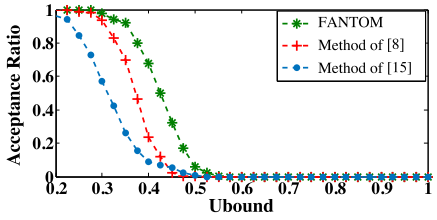

FIGURE 6. Acceptance ratio of FANTOM with $max(\delta) = 4$ in comparison with methods of [8] and [15].

3) EFFECTS OF USING FAULT-TOLERANCE TECHNIQUES

Now, we compare our proposed fault-tolerant scheme with the case when there is no fault-tolerant mechanism in the system. Indeed, we compare to a traditional non-MC scheduling algorithm in which the regular EDF algorithm is presented and applied in many previous studies [50]. Using the fault tolerance techniques such as re-execution to guarantee the system's reliability and safety requirement has timing overheads, which is a common practice [46]. However, since the MC systems are safety-critical, its correct operation throughout a complete time interval is crucial even in the case of fault occurrence to prevent catastrophic consequences [51]. Fig. 7 depicts that the proposed approach preserves the PFH of MCTs, and HCTs, to less than 10^{-7} and 10^{-9} (introduced in Table. 1), respectively, in any U_{bound} that the system is schedulable. In comparison, the traditional scheme has severely damaged the system's safety, which is not desirable. It should be noted that, as mentioned in subsection VI-B2, the system is not schedulable for $U_{bound} \geq 0.6$. Therefore, the results of the PFH for both MCTs and HCTs are not shown for $U_{bound} \geq 0.6$.

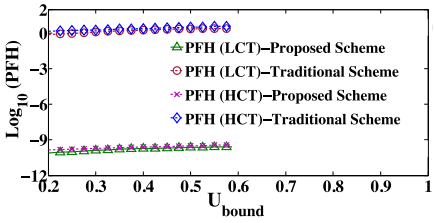

FIGURE 7. Safety requirement guarantee for the system with and without fault consideration.

4) EFFECTS OF HCT RUN-TIME BEHAVIORS ($P(C^{LO})$)

In this subsection, we evaluate the effect of changing the run-time behaviors of HCTs on the acceptance ratio. $P(C^{LO})$ denotes the probability that HCTs execute with their WCET in the LO mode (C_i^{LO}) (As discussed before, HCTs may overrun and use their WCET in the HI mode). It can be seen that if the inequalities (11) and (20) are satisfied offline, the task set will be schedulable in both criticality modes. Hence, the schedulability of the task set is not affected by the variation of $P(C^{LO})$ in run-time.

5) EFFECT OF VARYING PoF FOR TASK INSTANCES

We evaluate the impact of varying PoF (f) on the system schedulability. Here, we assumed that 40% of tasks are HCTs and 70% of the tasks are LCTs (30% MCT with $1 < \delta \leq 4$, and 30% non-MCTs with $\delta = 1$). As shown in Fig. 8, the acceptance ratio increases as (f) decreases from 10^{-5} to 10^{-7} and also from 10^{-7} to 10^{-9} in both our proposed method and the method proposed in [8]. The reason is that decreasing f means using a more reliable platform to have a safer system. However, the acceptance ratio of our proposed method is always better than the result of the proposed method in [8].

6) EFFECTS OF TASK MIXTURES WITH VARYING P(HCT) AND P(MCT)

Now, we evaluate the effect of HCT distribution variation on the acceptance ratio. $P(HI)$ denotes the ratio of HCTs to all

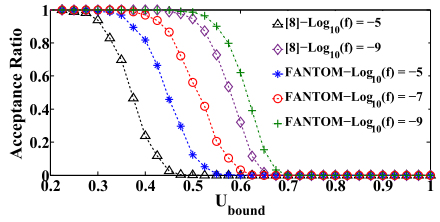

FIGURE 8. Acceptance ratio with varying the PoF, and $max(\delta) = 4$.

of the generated tasks. Here, in each scenario, we assume that the ratio of MCTs to all of the generated tasks is constant and the ratio of non-MCTs to all will be varied. Fig. 9 shows that the acceptance ratio improvement becomes pronounced when there are fewer HCTs in a task set (i.e., when P(HCT) is decreased). Because, when there are more HCTs in a task set, there will be less number of system switches to the HI mode and even by occurrence of a mode switch, the system will be switched back after a relatively short period of time. Consequently, the HI and LO modes will overlap less in time. In addition, LCTs would fail less. This reasoning can also be obtained by exploring the condition (20). Besides, the same trend is found for [8], except that the proposed schemes always perform better than [8].

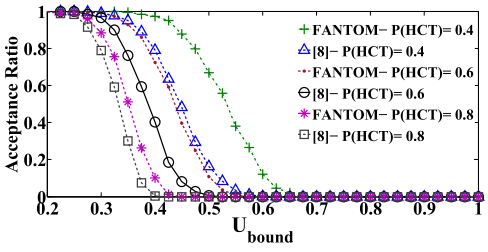

FIGURE 9. Acceptance ratio with varying *P(HCT)*.

Similar to the above case, in Fig. 10, we evaluate how the number of MCTs affects the acceptance ratio. We assume that the ratio of HCTs to LCTs is constant and the ratio of the MCTs to non MCTs in the task set is the only varying parameter. In addition, the distribution of HCTs in a task set has been considered as a constant value (0.3). According to equality (27), since the upper bound utilization may be influenced by the MCTs, the acceptance ratio will be changed. According to Fig. 10, it is evident that we would have a noticeable amount of improvement as the utilization of MCTs is reduced. Decreasing the number of MCTs in a task set can cause a reduction of utilization and consequently, more tasks can be scheduled.

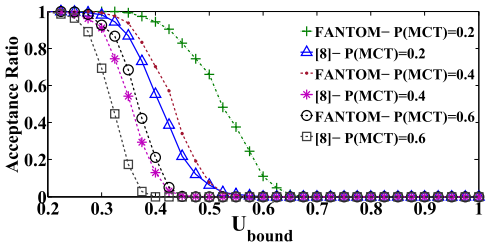

FIGURE 10. Acceptance ratio with varying *P(MCT)*.

To this end, based on the results, we can conclude that by minimizing the ratio of HCTs and MCTs in a task set, the upper bound will be maximized. In addition, the acceptance ratio of schedulable tasks would be increased by decreasing the task's PoF (i.e., more reliable hardware platform is used).

VII. CONCLUSION

In this article, we presented a new Mixed-Criticality (MC) task model and then analyzed task-drop aware scheduling for uni-processor MC systems and also, guarantee the safety requirements of MC tasks in the presence of faults. Existing tasks in these systems have different criticality levels from both real-time and safety perspectives. In some MC systems, some low-criticality tasks should not be frequently dropped in the High-Criticality (HI) mode to prevent catastrophic consequences. Therefore, by defining a new parameter that is determined by designers, we propose a task-drop aware scheduling analysis based on the EDF-VD to schedule both types of tasks in the HI mode. This parameter specifies the minimum interval between two consecutive drops in this mode. We analyzed the results by varying different parameters in the system, and obtained that the proposed method improves the acceptance ratio by up to 43.9% compared to state-of-the-art.

As future research, we would consider multi-core systems to use different fault tolerance techniques and more efficient MC task scheduling algorithms and obtain better performance. In addition, we will evaluate the method on a real platform.

APPENDIX A
PROOF OF EQUATION (15)

As we mentioned, there are some scenarios in which, the inequality (10) has been satisfied while HCTs in the HI mode have missed their deadline. The other condition to guarantee the scedulability of all HCTs and most of LCTs by our policy in their specified deadlines by the EDF-VD in the HI mode (Eq. 15), is as follows:

$$U_{HCT}^{HI} + (1 - x) \times U_{LCT}^{HI} + x \times (U_{LCT}^{LO}) \leq 1$$

Proof: To prove this inequality, suppose that τ_1 is an HCT with release time a_1 and deadline d_1 and causes the system switches to the HI mode at time t_1. Besides, τ_2 is an HCT that its deadline is missed at time t_2 while the system is in the HI mode ($0 < t_1 < t_2$). In addition, suppose that η_i is the cumulative execution time of each task τ_i in $[0, t_2]$. By this definition, we nominate $t_1 < (a_1 + x(t_2 - a_1))$. To prove this, consider that the absolute deadline of the HCT τ_1 is d_1 and its virtual deadline is $(a_1 + x(d_1 - a_1))$. As can be seen, τ_1 is overrun at time t_1 and continue to finish its execution completely before its deadline d_1, which is

less than t_2 ($d_1 \leq t_2$). Thus:

$$\implies t_1 < (a_1 + x(d_1 - a_1)) < (a_1 + x(t_2 - a_1)).$$

As we proposed, when the system switches to the HI mode, we drop the first job of each LCT τ_i and then, they would be dropped every δ_i times as long as the system is in the HI mode. If an LCT with release time a_i and deadline d_i, is released before time t_1, while its deadline d_i is after t_1 ($a_i < t_1 < d_i$), it is called carry-over job of LCTs. For these carry-over jobs, we have, $d_i < (a_1 + x(t_2 - a_1)$. To prove, it is obvious that the maximum cumulative execution time of an LCT τ_i is $(d_i - a_i)u_i^{LO}$ and it happens when the task can finish its execution before t_1. It means before overrunning of HCT τ_1. Since the EDF-VD algorithm is used for task scheduling, then, $d_i \leq x \times d_1 \implies d_i \leq a_1 + x(d_1 - a_1)$. Since we have $d_1 < t_2$, therefore:

$$\implies d_i \leq a_1 + x(t_2 - a_1).$$

Lemma: For any LCT τ_i:

$$\eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO} + (1 - x)(t_2 - a_1) \times u_i^{HI}.$$

To prove, let's consider two cases; LCT τ_i is released within interval $(t_1, t_2]$ or it is not.

- Case1 (task τ_i is released within the time interval $(t_1, t_2]$):
  - *With carry-over job ($t_1 < d_i$):* The maximum cumulative execution time of LCT τ_i within the time interval $[0, t_2]$ is $\eta_i \leq (a_i - 0) \times u_i^{LO} + (d_i - a_i) \times u_i^{LO} + (t_2 - d_i) \times u_i^{HI} \implies \eta_i \leq d_i \times u_i^{LO} + (t_2 - d_i) \times u_i^{HI}$.
  Since, we mentioned above, $d_i < (a_1 + x(t_2 - a_1))$, then $\eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO} + (t_2 - (a_1 + x(t_2 - a_1))) \times u_i^{HI} \implies \eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO} + (1 - x)(t_2 - a_1) \times u_i^{HI}$
  - *No carry-over job ($d_i < t_1$):* The maximum cumulative execution time of LCT τ_i within the time interval $[0, t_2]$ is $\eta_i \leq (t_1 - 0) \times u_i^{LO} + (t_2 - t_1) \times u_i^{HI}$.
  We mentioned that $t_1 < (a_1 + x(t_2 - a_1))$. Therefore, $\eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO} + (t_2 - (a_1 + x(t_2 - a_1))) \times u_i^{HI} \implies \eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO} + (1 - x)(t_2 - a_1) \times u_i^{HI}$
- Case 2 (task τ_i is not released within the time interval $(t_1, t_2]$): It means, the maximum cumulative execution of these tasks is $\eta_i \leq (d_i - 0) \times u_i^{LO}$. Now, we have two cases, $d_i \leq t_1$ and $d_i > t_1$. for the first one, since we mentioned, $t_1 < (a_1 + x(t_2 - a_1))$, and as we know that $d_i \leq t_1$, then $\eta_i < t_1 \times u_i^{LO} \implies \eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO} \implies \eta_i < (a_1 + x(t_2 - a_1)) \times u_i^{LO} + (1 - x)(t_2 - a_1) \times u_i^{HI}$. Now, for the case of $d_i > t_1$, we proved that $d_i < (a_1 + x(t_2 - a_1))$. Thus: $\eta_i < d_i \times u_i^{LO} \implies \eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO} \implies \eta_i < (a_1 + x(t_2 - a_1)) \times u_i^{LO} + (1 - x)(t_2 - a_1) \times u_i^{HI}$
  Since we calculate the cumulative execution of tasks in the time interval $[0, t_2]$, there are some LCTs with $\delta = 1$, which are executed in the time interval of $[0, t_1]$. Therefore, the maximum cumulative execution of these tasks is $\eta_i \leq (t_1 - 0) \times u_i^{LO}$. Since $t_1 < (a_1 + x(t_2 - a_1))$, then $\eta_i \leq (a_1 + x(t_2 - a_1)) \times u_i^{LO}$.

In addition, the maximum cumulative execution of HCTs in the time interval $[0, t_2]$ can be computed as $\eta_i \leq \frac{a_1}{x} \times u_i^{LO} + (t_2 - a_1) \times u_i^{HI}$. It should be mentioned that, HCTs are executed with their virtual deadline in the time interval $[0, a_1]$ and since, the HCT τ_1 overrun and the system switches to the HI mode, all tasks after a_1 will be executed by their actual deadline.

Now, let H denotes the cumulative execution of all tasks in the time interval $[0, t_2]$. Thus,

$$\begin{aligned}
H \quad \leq \quad & \sum_{\zeta_i \in LCTs \ \& \ \delta=1} (a_1 + x(t_2 - a_1)) \times u_i^{LO} \\
& + \sum_{\zeta_i \in LCTs \ \& \ \delta>1} (a_1 + x(t_2 - a_1)) \times u_i^{LO} \\
& + (1 - x)(t_2 - a_1) \times u_i^{HI} \\
& + \sum_{\delta_i \in HCTs} \frac{a_1}{x} \times u_i^{LO} + (t_2 - a_1) \times u_i^{HI} \\
\implies H \leq \ & (a_1 + x(t_2 - a_1)) \times U_{LCT}^{LO}|^{\delta=1} \\
& + (a_1 + x(t_2 - a_1)) \times U_{LCT}^{LO}|^{\delta>1} + (1 - x)(t_2 - a_1) \\
& \times U_{LCT}^{HI} + \frac{a_1}{x} \times U_{HCT}^{LO} + (t_2 - a_1) \times U_{HCT}^{HI} \\
\implies H \leq \ & (a_1 + x(t_2 - a_1)) \times U_{LCT}^{LO} + (1 - x)(t_2 - a_1) \\
& \times U_{LCT}^{HI} + \frac{a_1}{x} \times U_{HCT}^{LO} + (t_2 - a_1) \times U_{HCT}^{HI} \\
\implies H \leq \ & a_1 \times (U_{LCT}^{LO} + \frac{U_{HCT}^{LO}}{x}) + x(t_2 - a_1) \times U_{LCT}^{LO} \\
& + (1 - x)(t_2 - a_1) \times U_{LCT}^{HI} + (t_2 - a_1) \times U_{HCT}^{HI}
\end{aligned}$$

As presented in Eq. 10, $(U_{LCT}^{LO} + \frac{U_{HCT}^{LO}}{x}) \leq 1$, then:

$$\begin{aligned}
H \leq a_1 + x(t_2 - a_1) \times U_{LCT}^{LO} + (1 - x)(t_2 - a_1) \times U_{LCT}^{HI} \\
+ (t_2 - a_1) \times U_{HCT}^{HI}
\end{aligned}$$

Hence, H is the maximum cumulative execution of all tasks. As we mentioned, τ_2 is one of these tasks that its deadline is missed. Therefore, H would be greater than t_2 (the time, which τ_2 misses its deadline).

$$\begin{aligned}
& a_1 + x(t_2 - a_1) \times U_{LCT}^{LO} + (1 - x)(t_2 - a_1) \times U_{LCT}^{HI} \\
& \quad + (t_2 - a_1) \times U_{HCT}^{HI} > t_2 \implies x(t_2 - a_1) \\
& \quad \times U_{LCT}^{LO} + (1 - x)(t_2 - a_1) \times U_{LCT}^{HI} \\
& \quad + (t_2 - a_1) \times U_{HCT}^{HI} > t_2 - a_1 \\
& \quad \implies x \times U_{LCT}^{LO} + (1 - x) \times U_{LCT}^{HI} + U_{HCT}^{HI} > 1
\end{aligned}$$

Therefore, we must have the following inequality to guarantee that no HCT misses its deadline in the HI mode.

$$x \times U_{LCT}^{LO} + (1 - x) \times U_{LCT}^{HI} + U_{HCT}^{HI} \leq 1$$

REFERENCES

[1] A. Burns and R. Davis, "Mixed criticality systems-a review," Dept. Comput. Sci., Univ. York, York, U.K., Tech. Rep, 2013, pp. 1–69.
[2] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2013, pp. 147–152.
[3] M. Zimmer, D. Broman, C. Shaver, and E. A. Lee, "FlexPRET: A processor platform for mixed-criticality systems," in *Proc. IEEE 19th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2014, pp. 101–110.

[4] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, "Scheduling real-time mixed-criticality jobs," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1140–1152, Aug. 2012.

[5] S. Baruah, H. Li, and L. Stougie, "Towards the design of certifiable mixed-criticality systems," in *Proc. 16th IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2010, pp. 13–22.

[6] L. A. Johnson, "Do-178b, software considerations in airborne systems and equipment certification," *Crosstalk*, vol. 199, pp. 11–20, Oct. 1998.

[7] S. Baruah, V. Bonifaci, G. DAngelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Proc. 24th Euromicro Conf. Real-Time Syst.*, Jul. 2012, pp. 145–154.

[8] P. Huang, H. Yang, and L. Thiele, "On the scheduling of fault-tolerant mixed-criticality systems," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.

[9] J. H. Anderson, S. K. Baruah, and B. B. Brandenburg, "Multicore operating-system support for mixed criticality," in *Proc. Workshop Mixed Criticality, Roadmap Evolving UAV Certification*, vol. 4, 2009, p. 7.

[10] S. Baruah, V. Bonifaci, G. D'angelo, H. Li, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems," *J. ACM*, vol. 62, no. 2, pp. 14:1–14:33, 2015.

[11] L. Zeng, P. Huang, and L. Thiele, "Towards the design of fault-tolerant mixed-criticality systems on multicores," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*, 2016, pp. 6:1–6:10.

[12] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 239–243.

[13] B. Ranjbar, T. D. A. Nguyen, A. Ejlali, and A. Kumar, "Online peak power and maximum temperature management in multi-core mixed-criticality embedded systems," in *Proc. 22nd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2019, pp. 546–553.

[14] S. K. Baruah, V. Bonifaci, G. d'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Mixed-criticality scheduling of sporadic task systems," in *Proc. Eur. Symp. Algorithms*, 2011, pp. 555–566.

[15] Z. Al-bayati, J. Caplan, B. H. Meyer, and H. Zeng, "A four-mode model for efficient fault-tolerant mixed-criticality systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2016, pp. 97–102.

[16] A. Esper, G. Nelissen, V. Nélis, and E. Tovar, "How realistic is the mixed-criticality real-time system model?" in *Proc. 23rd Int. Conf. Real Time Netw. Syst. (RTNS)*, 2015, pp. 139–148.

[17] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran, "Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling," in *Proc. 25th Int. Conf. Real-Time Netw. Syst.*, Oct. 2017, pp. 237–246.

[18] O. Gettings, S. Quinton, and R. I. Davis, "Mixed criticality systems with weakly-hard constraints," in *Proc. 23rd Int. Conf. Real Time Netw. Syst. (RTNS)*, 2015, pp. 237–246.

[19] J. Lee, H. S. Chwa, L. T. X. Phan, I. Shin, and I. Lee, "Mc-adapt: Adaptive task dropping in mixed-criticality scheduling," *ACM Tran. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 163:1–163:21, 2017.

[20] D. Liu, N. Guan, J. Spasic, G. Chen, S. Liu, T. Stefanov, and W. Yi, "Scheduling analysis of imprecise mixed-criticality real-time tasks," *IEEE Trans. Comput.*, vol. 67, no. 7, pp. 975–991, Jul. 2018.

[21] Z. Guo, K. Yang, S. Vaidhun, S. Arefin, S. K. Das, and H. Xiong, "Uniprocessor mixed-criticality scheduling with graceful degradation by completion rate," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2018, pp. 373–383.

[22] H. Su, N. Guan, and D. Zhu, "Service guarantee exploration for mixed-criticality systems," in *Proc. IEEE 20th Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2014, pp. 1–10.

[23] V. K. Sundar and A. Easwaran, "A practical degradation model for mixed-criticality systems," in *Proc. IEEE 22nd Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2019, pp. 171–180.

[24] G. Chen, N. Guan, D. Liu, Q. He, K. Huang, T. Stefanov, and W. Yi, "Utilization-based scheduling of flexible mixed-criticality real-time tasks," *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 543–558, Apr. 2018.

[25] P. Huang, P. Kumar, G. Giannopoulou, and L. Thiele, "Run and be safe: Mixed-criticality scheduling with temporary processor speedup," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2015, pp. 1329–1334.

[26] J. Boudjadar, S. Ramanathan, A. Easwaran, and U. Nyman, "Combining task-level and system-level scheduling modes for mixed criticality systems," in *Proc. IEEE/ACM 23rd Int. Symp. Distrib. Simul. Real Time Appl. (DS-RT)*, Oct. 2019, pp. 1–10.

[27] K. Yang and Z. Guo, "EDF-based mixed-criticality scheduling with graceful degradation by bounded lateness," in *Proc. IEEE 25th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2019, pp. 1–6.

[28] D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, and W. Yi, "EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Nov. 2016, pp. 35–46.

[29] R. I. Davis, A. Zabos, and A. Burns, "Efficient exact schedulability tests for fixed priority real-time systems," *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1261–1276, Sep. 2008.

[30] R. I. Davis and M. Bertogna, "Optimal fixed priority scheduling with deferred pre-emption," in *Proc. IEEE 33rd Real-Time Syst. Symp.*, Dec. 2012, pp. 39–50.

[31] A. Burns and R. I. Davis, "Adaptive mixed criticality scheduling with deferred preemption," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 2014, pp. 21–30.

[32] J. Lin, A. M. Cheng, D. Steel, and M. Y.-C. Wu, "Scheduling mixed-criticality real-time tasks with fault tolerance," in *Proc. Workshop Mixed Criticality Syst.*, 2014, pp. 1–6.

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

[34] J. Lin, A. M. K. Cheng, D. Steel, M. Y.-C. Wu, and N. Sun, "Scheduling mixed-criticality real-time tasks in a fault-tolerant system," *Int. J. Embedded Real-Time Commun. Syst.*, vol. 6, no. 2, pp. 65–86, Apr. 2015.

[35] J. Zhou, M. Yin, Z. Li, K. Cao, J. Yan, T. Wei, M. Chen, and X. Fu, "Fault-tolerant task scheduling for mixed-criticality real-time systems," *J. Circuits, Syst. Comput.*, vol. 26, no. 01, Jan. 2017, Art. no. 1750016.

[36] A. Thekkilakattil, R. Dobrin, and S. Punnekkat, "Fault tolerant scheduling of mixed criticality real-time tasks under error bursts," *Procedia Comput. Sci.*, vol. 46, pp. 1148–1155, 2015.

[37] J. Caplan, Z. Al-bayati, H. Zeng, and B. H. Meyer, "Mapping and scheduling mixed-criticality systems with on-demand redundancy," *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 582–588, Apr. 2018.

[38] C. Deng, W.-W. Che, and P. Shi, "Cooperative fault-tolerant output regulation for multiagent systems by distributed learning control approach," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, 2020.

[39] J. Jiang and X. Yu, "Fault-tolerant control systems: A comparative study between active and passive approaches," *Annu. Rev. Control*, vol. 36, no. 1, pp. 60–72, Apr. 2012.

[40] B. Wang, B. Zhang, and R. Su, "Optimal tracking cooperative control for cyber-physical systems: Dynamic fault tolerant control and resilient management," *IEEE Trans. Ind. Informat.*, early access, Jan. 10, 2020, doi: 10.1109/TII.2020.2965538.

[41] R. Ernst and M. Di Natale, "Mixed criticality systems—A history of misconceptions?" *IEEE Des. Test.*, vol. 33, no. 5, pp. 65–74, Oct. 2016.

[42] M. Jafari-Nodoushan, B. Safaei, A. Ejlali, and J.-J. Chen, "Leakage-aware battery lifetime analysis using the calculus of variations," *IEEE Trans. Circuits Syst. I, Reg. Papers*, early access, Jun. 16, 2020, doi: 10.1109/TCSI.2020.3001064.

[43] *Road Vehicles—Functional Safety*, ISO Standard 26262, Dec. 2018.

[44] M. A. Awan, D. Masson, and E. Tovar, "Energy-aware task allocation onto unrelated heterogeneous multicore platform for mixed criticality systems," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 2015, p. 377.

[45] R. Narimani, B. Safaei, and A. Ejlali, "A comprehensive analysis on the resilience of adiabatic logic families against transient faults," *Integration*, vol. 72, pp. 183–193, May 2020.

[46] I. Koren and C. M. Krishna, *Fault-Tolerant System*. San Mateo, CA, USA: Morgan Kaufmann, 2007.

[47] E. Dubrova, "Fundamentals of dependability," in *Fault-Tolerant Design*. Cham, Switzerland: Springer, 2013, pp. 5–20.

[48] S. S. Sahoo, B. Veeravalli, and A. Kumar, "Cross-layer fault-tolerant design of real-time systems," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Sep. 2016, pp. 63–68.

[49] P. Ittershagen, K. Gruttner, and W. Nebel, "Mixed-criticality system modelling with dynamic execution mode switching," in *Proc. Forum Specification Design Lang. (FDL)*, 2015, pp. 1–6.

[50] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, vol. 24. Cham, Switzerland: Springer, 2011.

[51] A. Taherin, M. Salehi, and A. Ejlali, "Reliability-aware energy management in mixed-criticality systems," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 3, pp. 195–208, Jul. 2018.

BEHNAZ RANJBAR received the B.S. degree in computer engineering from the Amirkabir University of Technology, in 2012, and the M.S. degree from the Sharif University of Technology, Tehran, Iran, in 2014. She is currently pursuing the joint Ph.D. degree with the Sharif University of Technology and the Chair for Processor Design, Technische Universität (TU) Dresden, Dresden, Germany. Her research interests include real-time and low-power embedded system design.

BARDIA SAFAEI (Student Member, IEEE) is currently pursuing the Ph.D. degree in computer engineering with the Sharif University of Technology. He is also conducting research with the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany. His research interests include power efficiency and dependability challenges in the Internet of Things (IoT), cyber-physical systems (CPSs), and embedded systems. He has been honored as a member of the National Elites Foundation, since 2016. He was nominated for the Best Paper Award at ICSEC'16. He received the ACM/SIGAPP Student Award at the 34th ACM/SIGAPP Symposium on Applied Computing (SAC'19). He has served as a reviewer for several prestigious international journals and conferences.

ALIREZA EJLALI received the Ph.D. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2006. From 2005 to 2006, he was a Visiting Researcher with the Electronic Systems Design Group, University of Southampton, Southampton, U.K. He is currently an Associate Professor of Computer Engineering with the Sharif University of Technology, where he is also the Director of the Embedded Systems Research Laboratory, Department of Computer Engineering. His research interests include low-power design, real-time systems, and fault-tolerant embedded systems.

AKASH KUMAR (Senior Member, IEEE) received the joint Ph.D. degree in electrical engineering and embedded systems from the Eindhoven University of Technology, Eindhoven, The Netherlands, and the National University of Singapore (NUS), Singapore, in 2009. From 2009 to 2015, he was with NUS. He is currently a Professor with Technische Universität Dresden, Dresden, Germany, where he is also directing the Chair for Processor Design. His current research interests include the design, analysis, and resource management of low-power and fault-tolerant embedded multiprocessor systems.