BOT-MICS: Bounding Time Using Analytics in Mixed-Criticality Systems

Behnaz Ranjbar, Ali Hosseinghorban, Siva Satyendra Sahoo, Alireza Ejlali, and Akash Kumar, *Senior Member, IEEE*

***Abstract*—An increasing trend for reducing cost, space, and weight leads to modern embedded systems that execute multiple tasks with different criticality levels on a common hardware platform while guaranteeing a safe operation. In such Mixed-Criticality (MC) systems, multiple Worst-Case Execution Times (WCETs) are defined for each task, corresponding to system operation mode to improve the MC system's timing behavior at run-time. Determining the appropriate WCETs for lower criticality modes is non-trivial. On the one hand, considering a very low WCET for tasks can improve the processor utilization by scheduling more tasks in that mode, on the other hand, using a larger WCET ensures that the mode switches (which causes by task overrunning) are minimized, thereby improving the quality-of-service for all tasks, albeit at the cost of processor utilization. Hitherto, no analytical solutions are proposed to determine WCETs in lower criticality modes. In this regard, we propose a scheme to determine WCETs by Chebyshev theorem, to make a trade-off between the number of scheduled tasks at design-time and the number of dropped low-criticality tasks at run-time as a result of frequent mode switches. To have a tight bound of execution times and mode switching probability, we also propose a distribution analytics-based scheme, in which the mode switching probability is obtained based on the cumulative distribution function. Our experimental results show that our scheme improves the utilization of state-of-the-art MC systems by up to 72.27%, while maintaining 24.28% mode switching probability in the worst-case scenario. Besides, the results of running embedded real-time benchmarks on a real platform show that the distribution-based scheme can improve the utilization by 7.30% while bounding the mode switching probability by 4.85% more, compared to the Chebyshev-based scheme.**

***Index Terms*—Mixed-Criticality, Mode Switching Probability, Optimization, Resource Utilization, Schedulability, WCETs' Analysis, Tight Execution Time Bound.**

I. Introduction

NOWADAYS, implementing a complex system, executing various applications with different levels of assurance, is a growing trend in modern embedded real-time systems, to meet cost, space, timing, and power consumption requirements [1]–[5]. Medical devices, automotive, and avionics industries are the most common applications, exploiting these systems, which are known as Mixed-Criticality (MC) systems [6]. For instance, DO-178B [7] is an industrial standard that is used in the avionics industry and has introduced different levels of safety, in which, a failure/deadline miss in tasks with various criticality levels has a different impact on the system, from no impact to catastrophic consequences. Therefore, an efficient MC system design should be developed to guarantee the successful execution of all tasks with higher criticality (HC) level to prevent catastrophic damages while ensuring the efficient resource utilization and Quality-of-Service (QoS) maximization (i.e., execute a higher number of lower criticality (LC) tasks) [3], [4], [8].

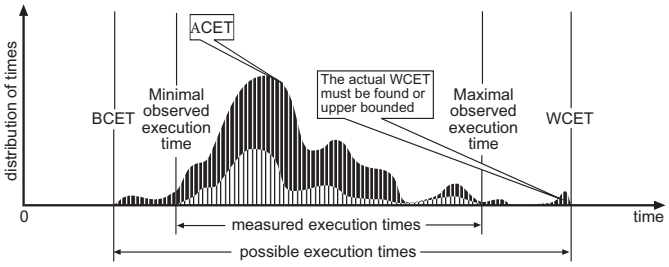


Fig. 1. Execution time distribution for a real-time task [9]. The figure shows the large gap between the WCET and the ACET.

In conventional real-time systems, the tasks are scheduled based on their pessimistic Worst-Case Execution Time (WCET). Many approaches like those presented in [9], [10] and tools like OTAWA [11] are used to determine the pessimistic WCET of a task by analyzing the task's control flow graph. These tools provide a safe and conservative execution time-bound so that no task's execution time exceeds the WCET under any circumstances. However, Fig. 1 [9] shows an execution time distribution of a task and observes that most samples' execution time is significantly shorter than such a conservative WCET. As a result, the resources would be severely under-utilized at run-time, which leads to poor processor utilization and QoS in conventional real-time systems.

To this end, in MC systems, tasks are analyzed with optimistic and pessimistic assumptions to obtain multiple WCETs, corresponding to the multiple criticality levels and the operation mode of the system [2], [12]–[14]. This ensures

Manuscript received June 25, 2021; revised September 27, 2021; accepted November 7, 2021. This work was supported in part by the German Research Foundation (DFG) Within the Cluster of Excellence Center for Advancing Electronics Dresden at the Technische Universität Dresden. This article was recommended by Associate Editor C. L. Yang. (Corresponding author: Akash Kumar.)

Behnaz Ranjbar, and Ali Hosseinghorban are with the Chair of Processor Design, CFAED, Technische Universität (TU) Dresden, 01062 Dresden, Germany and also with the Department of Computer Engineering, Sharif University of Technology, Tehran 11365-11155, Iran. (e-mail: behnaz.ranjbar@tu-dresden.de, ali.hosseinghorban@mailbox.tu-dresden.de)

Siva Satyendra Sahoo, and Akash Kumar are with the Chair of Processor Design, CFAED, Technische Universität (TU) Dresden, 01062 Dresden, Germany. (e-mail: {siva_satyendra.sahoo, akash.kumar}@tu-dresden.de)

Alireza Ejlali is with the Department of Computer Engineering, Sharif University of Technology, Tehran 11365-11155, Iran. (email: ejlali@sharif.edu)

that the processor utilization (and correspondingly, the QoS) is maximized in the low-criticality mode (LO mode), while the guarantees are preserved in the high-criticality mode (HI mode). At first, tasks are scheduled based on the optimistic WCETs. At run-time, if the execution time of at least one HC task exceeds its optimistic WCET (a task overruns), the system switches to the HI mode, i.e., the *mode switch* occurs due to the HC tasks' overrunning. In HI mode, to guarantee the correct execution of HC tasks, the system switches to the second scheduling, where all HC tasks are scheduled based on their pessimistic WCETs and all or some LC tasks are dropped [1]–[4], [6], [14], [15]. Therefore, when the gap between the optimistic and pessimistic WCETs is large, more tasks, especially LC tasks, are guaranteed to be scheduled in a processor at design-time. However, it may cause frequent system mode switches and, consequently, drop more LC tasks at run-time due to inefficient, optimistic WCET determination for HC tasks. When this gap is small, fewer LC tasks are scheduled in the LO mode which under-utilizes the processor. Indeed, this is overly pessimistic because, as shown in Fig. 1, tasks would be executed with less likelihood up to observed or actual WCET.

Therefore, optimistic WCETs play an important role in designing efficient MC systems and improving the timing behavior of these systems. Most state-of-the-art research works [2], [4], [12], [13], [15] set optimistic WCETs as a percentage of the pessimistic WCETs. However, Fig. 1 shows that most tasks' execution time is close to Average-Case Execution Time (ACET) (we discuss more in detail in Section IV). Furthermore, most studies have not analyzed the probability of exceeding the optimistic WCETs in system design. Besides, some studies determine the optimistic WCETs at run-time based on the system behavior and overall processing requirements. However, since in embedded systems, tasks are known in advance, and no new task is scheduled in the system, using these methods leads to poor utilization at run-time and can only optimize the probability of mode switching.

To the best of our knowledge, there is no method to determine optimistic WCETs for MC tasks to provide a reasonable trade-off between the number of scheduled LC tasks at design-time and the probability of mode switching at run-time to improve the system utilization and QoS. This paper[1] first proposes a novel scheme based on the *Chebyshev theorem* [17] for MC systems to determine the appropriate optimistic WCETs for tasks. Chebyshev theorem provides a general bound for all tasks with any distribution, which is pessimistic. To this end, we propose a second approach to determine tighter execution time bounds for HC tasks. In this approach, we analyze the execution time distribution of each task and fit a known distribution curve to it. Then we use the Cumulative Distribution Function (CDF) of the known distribution to provide a tight bound for probability of task overrunning and consequently, determining the optimistic WCET for that task. This article focuses on MC systems with two criticality levels, but our scheme can be extended for MC systems with several criticality levels by determining different values of WCETs corresponding to different system modes and solve the optimization problem.

[1]This article is an extended version of the prior work [16], with the title 'Improving the Timing Behaviour of Mixed-Criticality Systems Using Chebyshev's Theorem', which has been published in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021.

Contributions: The main contributions of this paper are:

- Introducing a novel scheme to obtain the optimistic WCETs by *Chebyshev theorem* in MC systems and showing the relation between the optimistic WCETs and mode switching probability, that is proposed in [16].
- Determining the number of adequate samples for computing ACET and standard deviation.
- Representing the tighter execution time bound and more realistic overrunning probability based on the applications' distribution time feature.
- Formulating and solving an optimization problem for improving the resource utilization and reducing the mode switching probability using Genetic Algorithm (GA).
- Evaluating our proposed scheme for various state-of-the-art MC systems to investigate their timing behaviour with real benchmarks on a real board, ODROID XU4.

Organization: The rest of the paper is organized as follows. In Section II, we review the related works. In Section III, we introduce MC task model and system operational modes. The motivational example and our proposed method are presented in Sections IV and V, respectively. Finally, we analyze the experimental results in Section VI and conclude in Section VII.

II. Related Works

A significant number of papers have been published in the last decade regarding the design of MC systems. Since our focus is on improving the timing behaviour of these MC systems by defining suitable WCETs and analyzing the probability of mode switching based on these WCETs, we only consider the works presented for designing these systems with similar scope.

The MC task model has been presented by Vestal in [18] for the first time, and introduced different WCET levels for tasks. However, the author has not discussed how these WCETs are obtained and how often the system switches to the HI mode based on the design. The authors have discussed a bit further in [19] that how different WCETs can be defined and determined. As an example, they can be determined at different levels of accuracy with different degrees of confidence by limiting the programming constructs, used in implementing the task. However, this approach does not involve any analysis. Most of the approaches, such as [1], [2], [4], [12], [13], [15], [20]–[22], generally count the optimistic WCETs ($WCET^{opt}$) as a percentage of the pessimistic WCETs ($WCET^{pes}$). This policy may waste the system utilization, or cause frequent mode switches, which disturbs the LC tasks and reduces their QoS (more detail is discussed in Section IV). Although the efficiency of these approaches has been evaluated for different percentages of $WCET^{pes}$, there is no scalable approach for determining the WCETs for all criticality levels. A few studies [23], [24] have determined the $WCET^{opt}$ of tasks at run-time, based on their overall processing requirements and actual execution times. However, there is no guarantee at

design-time on optimal use of the system utilization and LC tasks' execution.

Besides, a few studies such as [25], [26], have focused on probability distributions in MC systems by exploiting Extreme Value Theory (EVT) [27] for timing analysis. Note that, EVT is a branch of statistics, which estimates and models the probability distribution of extreme events. In the field of real-time systems, EVT is exploited to determine WCETs. Applying these estimation methods has some open challenges, such as the required number of execution times for a sample and its incomplete representativity identification and evaluation, that make it uncertain and unreliable [28]–[30]. Researchers in [30] have recently exploited this probabilistic information and proposed a technique to optimize the energy consumption of MC systems by finding the optimum core speed in the LO mode and based on that, obtaining the $WCET^{opt}$. However, their system operation model definition for running the LC tasks is different from the popular MC model. In this system, all LC and HC tasks are executed in both LO and HI modes, and the authors have obtained the $WCET^{opt}$ for HC tasks to investigate the trade-off between the minimum core frequency (that leads to energy minimization) and probability of mode switching. Switching the system to the HI mode causes an increase in processor frequency to guarantee all task schedulability before their deadlines. In fact, although this method improves the energy consumption, it causes to schedule fewer LC tasks in the system which leads to under-utilization.

From the mode switching probability perspective, some research works such as [31], have addressed mode switching probability in MC systems and how to have the safe mode switching at run-time. However, the relation between the HC tasks' $WCET^{opt}$ and mode switching probability has not been discussed.

Therefore, an appropriate WCET analysis of MC tasks in LO mode is needed to reduce the use of WCET estimation methods and improve the confidence in the WCET's values [3]. In this work, we propose a scheme to not just determine the WCETs in the LO mode, but also exploit them to optimize the system utilization, schedulability, and mode switching probability.

III. Mixed-Criticality Task Model

We consider a dual-criticality system analogous to that of [1], [4], [12], [12], [30], in which multiple periodic tasks with two criticality levels are executed upon the same platform. Each system has a finite number of MC tasks, $\{\tau_1, \tau_2, ..., \tau_t\}$. We characterize a task τ_i as $(\zeta_i, C_i^{LO}, C_i^{HI}, P_i, D_i)$, where:

- ζ_i denotes the criticality level of τ_i ($\zeta_i \in [LC, HC]$).
- C_i^{LO} (C_i^{HI}) denotes the WCET of task τ_i in LO (HI) mode.
- P_i denotes the period of task τ_i, which is the minimum amount of the time between two released instances.
- D_i denotes the deadline of task τ_i, $D_i = P_i$ [2], [12].

We consider an independent periodic task model as a case study to analyze the task schedulability. Note that, our proposed scheme is not limited to independent periodic tasks, and it can be used for any MC task set regardless of the dependency between tasks. Further, since we have dual-criticality systems, we have two levels of WCET for each HC task τ_i where $C_i^{HI} \geq C_i^{LO}$, $C_i^{HI} = WCET_i^{pes}$, and $C_i^{LO} = WCET_i^{opt}$. Since we use the utilization bound to schedule the MC tasks, the utilization of task τ_i at criticality mode l is defined as $u_i^l = \frac{C_i^l}{P_i}$ and $l \in \{LO, HI\}$.

System Operation Model: At first, the system begins its operation in LO mode, where all tasks (LC and HC) are executed correctly before their deadlines. If the execution time of at least one HC task exceeds its lowest WCET (C_i^{LO}), the system switches to the HI mode, and all HC tasks continue their execution by their largest WCET (C_i^{HI}). In this mode, since the HC tasks are supposed to execute longer, compared to the LO mode, the LC tasks are degraded to guarantee the correct execution of HC tasks before their deadlines and prevent catastrophic consequences. The system switches back to LO mode if there is no 1) ongoing HC task, executing on the processor, 2) ready HC task in the processor's queue [1]–[4]. From the perspective of LC tasks' degradation in the HI mode, the system should execute these LC tasks to improve its QoS and functionality. Note that, the QoS can be defined as the percentage of executed LC tasks to all LC tasks in the HI mode [8], [32] ($QoS = n_{LC}^{executed}/n_{LC}^{total}$, where n_{LC}^{total} is the number of all LC tasks and $n_{LC}^{executed}$ is the number of executed LC tasks in the HI mode).

IV. Motivational Example

In this example, we executed 20000 instances of five real-world applications, and their ACETs and WCETs in terms of CPU clock cycle are presented in Table I. $WCET^{pes}$ of each application is determined by OTAWA [11]. For each application, Table I also shows how many instances violate their $WCET^{opt}$ when it is set to ACET, or fraction ($\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$) of the $WCET^{pes}$ [2], [4], [15]. The important point that the table shows, is by increasing the size of inputs to an application, the ACET and $WCET^{pes}$ growth are not the same. For instance, the growth of $WCET^{pes}$ and ACET for ‹qsort›, a known algorithm for sorting arrays, is O(k^2) and O($k \log k$), respectively, where k is the size of the input array. Therefore, the $WCET^{pes}$ of ‹qsort› application for three different array sizes with 10, 100, 10000 elements, are 8.1, 22.7, and 59.0 times higher than the ACET of them, respectively. This table shows that $WCET^{pes}$ is not an appropriate parameter to set $WCET^{opt}$. For example, by setting $WCET^{opt}$ to $\frac{WCET^{pes}}{16}$, the mode switching probability for ‹edge›, and ‹qsort-10› is more than 99%, while for ‹smooth›, ‹epic›, ‹qsort-100›, and ‹qsort-10000›, it is less than 2%. On the other hand, when the $WCET^{opt}$ is equal to ACET, the mode switching probability is between 43% to 55% for all applications. So, based on the results in Table I, we can conclude that the mode switching probability is more consistent when the $WCET^{opt}$ is estimated based on ACET, rather than $WCET^{pes}$. However, simply setting $WCET^{opt}$ equal to ACET leads to many system mode changes (almost half of the instances).

This paper introduces a scheme that provides a general formula to choose a suitable $WCET^{opt}$ based on ACET to

TABLE I
COMPARISON BETWEEN ACET AND WCET OF DIFFERENT APPLICATIONS

App	ACET (Cycle)	Pessimistic-WCET (Cycle)	Standard-Deviation (Cycle)	Percentage (%) of Samples that Overruns if the optimistic WCET is set to:					
				$ACET$	$\frac{WCET^{pes}}{4}$	$\frac{WCET^{pes}}{8}$	$\frac{WCET^{pes}}{16}$	$\frac{WCET^{pes}}{32}$	$\frac{WCET^{pes}}{64}$
qsort-10	2.3×10^2	1.9×10^3	3.9×10^1	50.52	0.00	45.52	99.98	100.00	100.00
qsort-100	1.8×10^4	4.1×10^5	1.2×10^3	50.22	0.00	0.02	0.02	99.98	99.98
qsort-10000	1.8×10^8	1.0×10^{10}	1.1×10^6	43.86	0.00	0.00	0.02	0.02	99.98
corner	5.6×10^5	9.4×10^6	6.2×10^4	53.27	0.00	0.00	47.71	100.00	100.00
edge	9.8×10^5	1.1×10^7	1.1×10^5	54.88	0.00	0.00	99.84	100.00	100.00
smooth	1.9×10^7	4.9×10^8	5.1×10^6	54.31	0.00	0.00	1.41	78.85	97.25
epic	1.1×10^7	7.0×10^8	1.9×10^6	52.85	0.00	0.00	0.00	0.00	52.20

improve the utilization of the system. This approach makes a reasonable trade-off between the mode switching probability and the time that a core becomes idle because of the gap between its actual execution time and the $WCET^{opt}$.

V. PROPOSED SCHEME

In this section, at first, we propose our scheme for determining the optimistic WCETs. We then present how to estimate ACET to be used in determining the WCETs in Section V-B. Further, a new scheme is proposed in Section V-C to determine a tighter execution time bound, compared to the first proposed scheme. At the end, we discuss the scheduling policy and optimization problem based on our new proposed schemes in Section V-D and V-E, respectively.

A. Determining optimistic WCET and overrunning probability

As mentioned earlier, determining the appropriate $WCET^{opt}$ for HC tasks is a major design challenge for MC systems. The proposed scheme designs the MC systems and analyzes the MC tasks of the application in the offline phase. Based on the analysis results, the scheme chooses a suitable $WCET^{opt}$ for each HC task based on their ACET, which improves the number of scheduled LC tasks due to the big gap between the ACET and WCET. To determine $WCET^{opt}$, we introduce the following theorem based on Chebyshev's theorem. Note that, Chebyshev's theorem is a technique for bounding a tail distribution, which is used for estimating the failure probability and also establishing high probability bounds. In fact, it determines where most of the data samples fall within a distribution. Note that this theorem disregards how the data are distributed. By knowing only the mean (ACET in this paper) and standard deviation of data samples, this theorem claims that a certain fraction of these data is less than a certain distance from the mean [33].

Theorem 1: Given a task τ_i, for any positive integer n, the rate at which the execution time exceeds the value $(ACET_i + n \times \sigma_i)$ for task τ_i is bounded with $\frac{1}{1+n^2}$.

Hence, by considering Chebyshev's theorem (presented below in detail), n can be any positive integer value. However, in our proposed method, it plays an important role to draw a trade-off between determining the $WCET^{opt}$ values and the probability of mode switching. We explain its role after formulating these two parameters.

Proof: We use Chebyshev's theorem to prove *Theorem 1*:

One-Sided Chebyshev *[33]*: For any non-negative random variable X, if $E[X]$ is the mean and $Var = \sigma^2$ is its variance, then, for any positive real number $a > 0$, we have the theorem (1):

$$Pr[(X - E[X]) \geq a] \leq \frac{\sigma^2}{\sigma^2 + a^2} \tag{1}$$

In this theorem, if a is equivalent to $n \times \sigma$ $(a \equiv n \times \sigma)$:

$$Pr[(X - E[X]) \geq n \times \sigma] \leq \frac{1}{1 + n^2} \tag{2}$$

Now, assuming m samples of task τ_i $(j_{i,1}, j_{i,2}, ..., j_{i,m})$ with execution time $C_{i,1}, C_{i,2}, ..., C_{i,m}$, the expected value $E[X]$ of task τ_i is:

$$E[X] = ACET_i = \frac{1}{m} \sum_{j=1}^{j=m} C_{i,j} \tag{3}$$

By using the expected value $ACET_i$ (we present how to compute $ACET$ for each task τ_i in Section V-B), the standard deviation of execution time, σ_i, for task τ_i is calculated as follows:

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{j=1}^{j=m} (C_{i,j} - ACET_i)^2} \tag{4}$$

If the execution time of a task is considered as a random variable, by using the *Chebyshev's theorem*, we can show that less than $\frac{1}{1+n^2}$ of samples have higher execution time than n standard deviation $(n \times \sigma)$ of the mean execution time ($ACET$=E[X]).

$$Pr[X \geq (ACET_i + n \times \sigma_i)] \leq \frac{1}{1 + n^2} \tag{5}$$

Therefore, the rate of exceeding the execution time level $(ACET_i + n \times \sigma_i)$ for task τ_i is bounded with $\frac{1}{1+n^2}$. ■

This theorem provides a general upper bound on the probability of exceeding any arbitrary execution time level for any task, independent of its distribution. To determine $WCET^{opt}$, *Chebyshev's theorem* can be applied, which requires mean ($ACET$, that we discuss later how to compute it in the next subsection) and standard deviation of the execution time (σ) of each task.

$$C_i^{LO} = WCET_i^{opt} = (ACET_i + n_i \times \sigma_i) \tag{6}$$

Parameter n should be set very carefully because a large value of n reduces the number of scheduled tasks in LO

mode, and a small n increases the probability of mode switching $P_i^{MS} = \frac{1}{1+n^2}$. In Section VI, we evaluate the impact of different values of n in computing the $WCET^{opt}$ and the probability of system mode switching (P_{sys}^{MS}).

In addition, since the value of $WCET^{opt}$ is based on the ACET, we need to calculate the average execution time for each task. In general, it is hard to achieve the real mean (μ) with all possible samples of tasks [34]; so, we discuss a method in the next subsection to estimate the empirical mean ($\hat{\mu}$) with the minimum number of samples.

B. ACET estimation and its minimum required samples

In this subsection, in order to calculate the $WCET_i^{opt}$ for each task τ_i, we explain how to estimate the $ACET_i$. Therefore, we need to determine how many samples (m) are required for ACET estimation. We present the estimation by the probability $1-\delta$ and ϵ error as follows.

Theorem 2: For any task τ_i, consider m as the number of samples; if $m \geq ln(\frac{2}{\delta})\frac{(WCET_i^{pes})^2}{2(\epsilon\times\mu)^2}$, there is an (ϵ,δ)-approximation for computing $ACET$ of task τ_i.

Proof: Considering m samples of task τ_i, where $C_{i,1}, C_{i,2}, ..., C_{i,m}$ are their execution times. Then, empirical mean ($\hat{\mu}$) for task τ_i is computed as Eq. (7).

$$\hat{\mu} = \frac{1}{m}\sum_{j=1}^{j=m} C_{i,j} \tag{7}$$

To prove *Theorem 2*, we use *Hoeffding Bound* theorem [17] to approximate the real mean (μ). Note that, *Hoeffding Bound* theorem provides an upper bound on the probability that the sum of random variables with a bounded range deviates from its expected value by more than a certain value [17], [35]. The execution time of a sample is an independent random variable because the execution time of one sample does not affect other samples' execution time.

Hoeffding Bound: Let $C_{i,1}, C_{i,2}, ..., C_{i,m}$ be independent random variables which are bounded by an interval $[a,b]$, then:

$$Pr[|\hat{\mu} - E[\hat{\mu}]| \geq \epsilon] \leq 2e^{\left(-\frac{2m\epsilon^2}{(b-a)^2}\right)} \tag{8}$$

The *Hoeffding* theorem bounds $Pr[|\hat{\mu}-\mu| \geq \epsilon]$ by using the fact that $E[\hat{\mu}] = \sum_{j=1}^{j=m} E[C_{i,j}] = \mu$. Thus, it can estimate the real mean μ with ϵ error. Based on the Hoeffding theorem, the execution time of each sample must be bounded by an interval $[a,b]$. The upper bound execution time of task's samples is the pessimistic WCET of that task ($WCET^{pes}$), so the execution time of each instance is bounded by $[0, WCET_i^{pes}]$ interval. Therefore, $b-a \leq WCET_i^{pes}$. If we consider $\epsilon = \epsilon^* \times \mu$, Eq. (8) is written as:

$$Pr[|\hat{\mu} - \mu| \geq \epsilon^* \times \mu] \leq 2e^{\left(-\frac{2m(\epsilon^*\times\mu)^2}{(WCET_i^{pes})^2}\right)} \tag{9}$$

In order to estimate the real mean with the minimum number of samples, we use a definition of (ϵ,δ)-*Approximation* [17].

(ϵ,δ)-Approximation: An algorithm gives an (ϵ,δ)-approximation for the input value V, if the output X of this algorithm, satisfies the following inequality. In fact, output X approximates input V with probability $1-\delta$ and ϵ error.

$$Pr[[X-V] \leq \epsilon V] \geq 1-\delta \Leftrightarrow Pr[[X-V] \geq \epsilon V] \leq \delta \tag{10}$$

By using this definition and Eq. (9), we present the following equation to achieve a $(1-\delta)$ confidence for the correctness of such an approximation:

$$2e^{\left(-\frac{2m(\epsilon\times\mu)^2}{(WCET_i^{pes})^2}\right)} \leq \delta \implies ln(\frac{2}{\delta})\frac{(WCET_i^{pes})^2}{2(\epsilon\times\mu)^2} \leq m \tag{11}$$

This equation shows that with $m \geq ln(\frac{2}{\delta})\frac{(WCET_i^{pes})^2}{2(\epsilon\times\mu)^2}$ instances, $\hat{\mu}$ is an (ϵ,δ)-approximation for μ.

$$Pr[|\hat{\mu}-\mu| \geq \epsilon\times\mu] \leq \delta \qquad \blacksquare \tag{12}$$

TABLE II
THE EFFECT OF VARYING n ON THE OVERRUNNING OF DIFFERENT TASKS FROM MIBENCH SUITE [36], UNDER THE PROPOSED CHEBYSHEV-BASED SCHEME AND EXPERIMENTS

	Chebyshev	bitcount	qsort	matrix-mult	smooth	corner
n=0	100.00%	43.31%	33.92%	42.33%	33.47%	7.96%
n=1	50.00%	8.87%	6.30%	16.26%	19.95%	4.95%
n=2	20.00%	3.68%	4.37%	4.18%	4.92%	3.98%
n=3	10.00%	0.92%	2.33%	0.91%	1.43%	3.08%
n=4	5.88%	0.71%	1.12%	0.22%	0.39%	2.22%

C. Determining a tight execution time bound

Eq. (5) presents a general theorem that is applied to any time distribution of tasks. Therefore, it might not provide a tight upper bound for the probability of mode switching. For example, if we consider $n = 0$ ($C_i^{LO} = ACET_i$), the rate of exceeding $ACET_i$ for task τ_i is bounded with 100% by Chebyshev theorem. It means the execution time of all samples of task τ_i might be more than $ACET_i$, which is not true for most distributions. Although it is not wrong, it does not provide a piece of useful information. Table II shows the percentage of overruns for five different applications, from MiBench suite [36] through experiments and our analysis, Chebyshev-based scheme. As shown, the proposed scheme can provide an upper bound which is valid for any execution time distribution. However, this scheme gives a pessimistic and loose upper bound for many applications. As an example, the percentage of overruns in experiments for ‹corner› application, is 7.96% when $n = 0$, while according to our scheme, it is estimated to be 100%.

Since in our case, the tasks' execution time distribution for some applications is known, we propose another scheme, alternative one, to determine the tighter execution time bounds. As we discuss further, the determined WCETs would be more realistic, which cause the method to be more scalable. Note that, this method might help for better scale to multiple criticality levels and thus, better management of mode switches. To preset the tighter execution bounds, we execute several benchmarks on a real board (we discuss the details in Section VI) and investigate their time distributions. Fig. 2a depicts the execution time distribution of four applications, from MiBench suite [36]. The distribution curve of these applications is very similar to existing known probability distributions. Therefore,

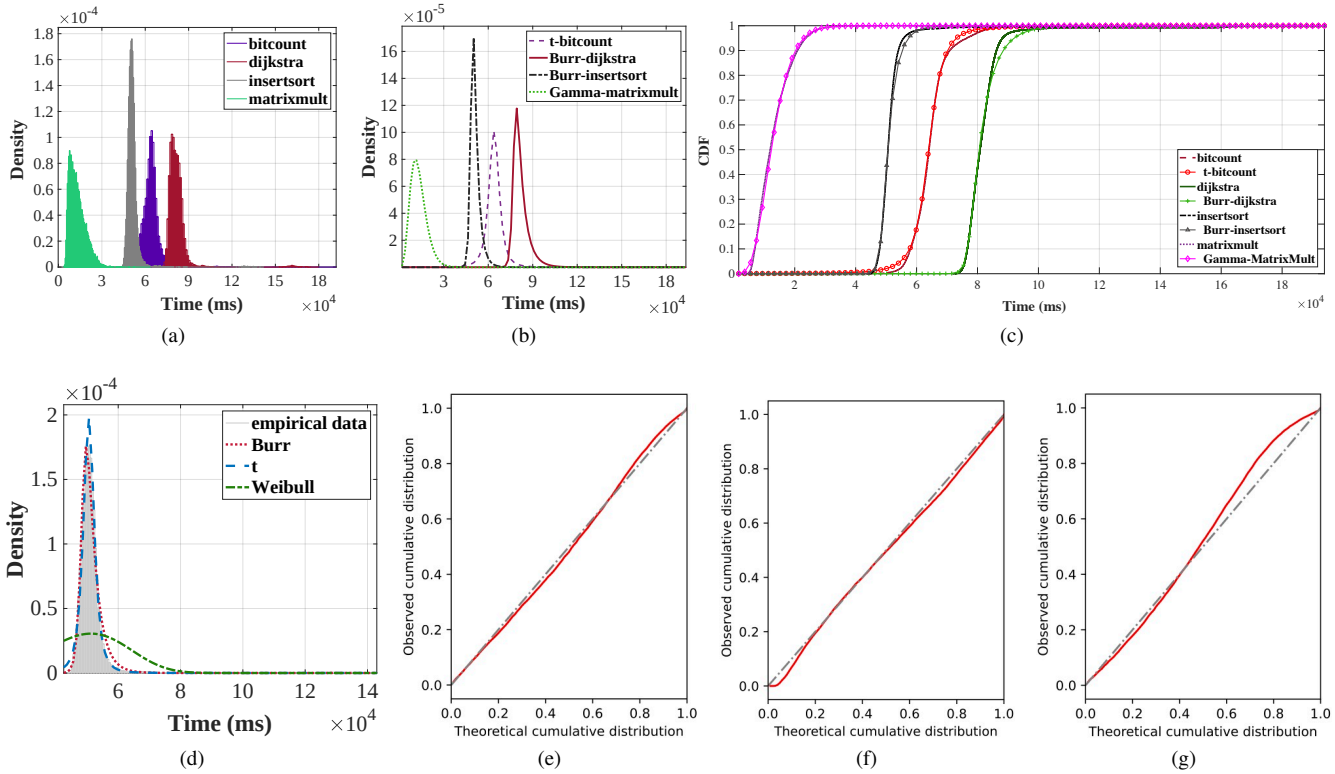


Fig. 2. Empirical execution time distributions and fitted distributions. (a) Empirical time distribution. (b) Fitted distribution. (c) Empirical and fitted CDF. (d) Top-3 distributions' PDF for insertsort. (e) CDF of Bur distribution. (f) CDF of t distribution. (g) CDF of Weibull distribution.

we fitted these applications with the well-known distributions. We used features of those distributions like probability density function (PDF) and cumulative distribution function (CDF) to estimate a tighter upper bound for mode switching. A distribution's CDF shows the probability that the execution time of an instance is less than or equal to a certain value, which we can consider as the optimistic WCET. Fig. 2b shows the fitted PDF and Fig. 2c shows the empirical and fitted CDF for four benchmarks. Since the probability of each task overrunning is important in our proposed method, we use the CDF formula (based on the best-fitted distribution) as $(1 - P_i^{MS})$ in our proposed method to find a tighter bound.

To identify the best-fitted distribution for the applications' execution time data, we have considered 16 different data distributions such as *Normal*, *Burr*, *Gamma*, *t*, *Weibull*, *Log-normal*, etc.. We evaluate the distributions' efficacy using Kolmogorov-Smirnov's (K-S) fitness metric [37], which is a commonly used technique. We select top-three distributions to implement the corresponding fitness functions for each application. As an example, Fig. 2d shows the density of top-3 distributions for the ‹insertion-sort› application, that are Burr, t, and Weibull distributions. Besides, to see how well a distribution fits data, we show how empirical data is distributed compared with a fitted distribution. Therefore, by using probability–probability (p-p) plot [38], we show two CDFs against each other. Fig. 2e, 2f, and 2g show it for the empirical and fitted data for top-three distributions. As shown, the Burr distribution (Fig. 2e) is more matched between the observed and theoretical cumulative distributions, compared to *t* distributions (Fig. 2f) and Weibull (Fig. 2g) distribution.

In the end, to compute a tighter probability of task overrunning ($Prob_i^{MS}$) based on n, $ACET$ and σ, instead of using Eq. (5), the CDF of the determined distribution ($F_i(t)$) is used as $Prob_i^{MS} = 1 - F_i(ACET_i + n \times \sigma_i)$.

D. Task schedulability analysis

In this subsection, we analyze the task schedulability and present the conditions based on the new formula of $WCET^{opt}$, determined in previous subsections. To schedule MC tasks in the uni-processor, we apply the existing MC scheduling technique, EDF-VD algorithm, which has been used in many studies since the last decade [1], [2], [4]. Here, when the system switches to the HI mode, all LC tasks are dropped. If U_l^k denotes total utilization of tasks with the same criticality level l in the mode k, then:

$$U_{HC}^{LO} = \sum_{\zeta_i = HC} \left(\frac{ACET_i + n_i \times \sigma_i}{T_i} \right) \quad and \quad U_{HC}^{HI} = \sum_{\zeta_i = HC} \frac{C_i^{HI}}{T_i} \tag{13}$$

A suitable $WCET^{opt}$ for each HC task τ_i can be achieved by choosing the optimum n_i (used in Eq. (6)). The optimum n_i must be determined to minimize the mode switching probability and maximize the resource utilization. To solve this, we formulate the optimization problem to find the optimum n_i for each task τ_i and determine its $WCET_i^{opt}$. Furthermore, Eq. (14) must be satisfied to guarantee schedulability by EDF-VD at run-time [1]. Eq. (14) presents the necessary and sufficient conditions to guarantee the task schedulability in

both LO and HI modes and meeting deadlines of running tasks even if the system switches to the HI mode [1].

$$U_{HC}^{LO} + U_{LC}^{LO} \leq 1 \quad and \quad U_{HC}^{HI} + \left(\frac{U_{HC}^{LO} \times U_{LC}^{LO}}{1 - U_{LC}^{LO}}\right) \leq 1 \quad (14)$$

E. Optimization problem formulation

In order to formulate the optimization problem based on the two objectives (mode switching probability and system utilization), we first identify the variables and constraints for better understanding. For each task τ_i, $ACET_i$, $WCET_i^{pes}$, T_i (period) and σ_i (standard variation) are constant. $WCET_i^{opt}$ is variable, which is computed based on the variable n_i (Introduced in Section V-A). These constant parameters and variables are used to compute the objectives, mode switching probability, and system utilization. In order to optimize these objectives and find the optimum value for n_i, we first present the constraints and then formulate the objectives as follows.

Execution Time Constraint: $WCET_i^{opt}$ of each HC task τ_i must not be more than $WCET_i^{pes}$.

$$(ACET_i + n_i \times \sigma_i) \leq WCET_i^{pes} \quad (15)$$

There are two main objectives to optimize the system:

***Objective 1**: Mode Switching Probability:* If the LC tasks are dropped frequently due to the HC tasks overrunning, it may negatively impact the performance or functionality of MC systems. Therefore, one of the most significant objectives is the minimization of mode switching probability. Let P_{Sys}^{MS} denote the probability of system mode switching. If P_{Sys}^{noMS} is the probability that no HC task overruns and consequently, no mode switch happens, then, $P_{Sys}^{MS} = 1 - P_{Sys}^{noMS}$. Since tasks are independent, P_{Sys}^{MS} is computed as shown in Eq. (16), where P_i^{MS} is the probability of task overrunning for task τ_i. According to our discussion in Section V-A, $P_i^{MS} = \frac{1}{1+n_i^2}$. The higher the n_i, the less the mode switching probability.

$$P_{Sys}^{MS} = 1 - \prod_{\zeta_i \in HC} (1 - P_i^{MS}) = 1 - \prod_{\zeta_i \in HC} \left(1 - \frac{1}{(1 + n_i^2)}\right) \quad (16)$$

***Objective 2**: Resource Utilization:* The second objective is to improve the resource utilization by a significant gain in terms of the utilization that can be allocated to LC tasks in the LO mode (U_{LC}^{LO}). Although maximizing U_{LC}^{LO} is desired, it is upper-bounded by the schedulability constraints, which can be derived from Eq. (14). Eq. (17) presents the condition to guarantee the task schedulability in the LO mode under the EDF-VD algorithm. In addition, as mentioned in Section V-D, Eq. (18) shows the condition for guaranteeing the task schedulability in the HI mode and mode switching [1], [8]. In these equations, the maximum amount of U_{LC}^{LO} depends on the values of n_i for each HC task. The lower the n_i, the higher the U_{LC}^{LO}. Therefore, the second objective can be bounded as follows.

$$U_{HC}^{LO} + U_{LC}^{LO} \leq 1 \implies$$
$$U_{LC}^{LO} \leq (1 - U_{HC}^{LO}) = 1 - \sum_{\zeta_i = HC} \left(\frac{ACET_i + n_i \times \sigma_i}{T_i}\right) \quad (17)$$

TABLE III
THE MINIMUM VALUE OF n IN $WCET_i^{opt} \geq WCET_i^{pes}$ FOR DIFFERENT TASKS

	FFT	qsort	dijkstra	corner	edge	smooth	epic	bitcount
n	60	17	12	11	27	8	7	19

$$U_{HC}^{HI} + \left(\frac{U_{HC}^{LO} \times U_{LC}^{LO}}{1 - U_{LC}^{LO}}\right) \leq 1 \implies U_{LC}^{LO} \leq \left(\frac{1 - U_{HC}^{HI}}{1 - U_{HC}^{HI} + U_{HC}^{LO}}\right)$$
$$\implies U_{LC}^{LO} \leq \left(\frac{1 - U_{HC}^{HI}}{1 - U_{HC}^{HI} + \left(\sum_{\zeta_i = HC} \left(\frac{ACET_i + n_i \times \sigma_i}{T_i}\right)\right)}\right) \quad (18)$$

Hence, if P_{Sys}^{MS}=1, it means the system is always in the HI mode, and all LC tasks are always dropped. If P_{Sys}^{MS}=0, it implies all LC tasks are always executed with no dropping. Therefore, by having these two objectives, we maximize the following equation.

$$maximize\{(1 - P_{Sys}^{MS}) \times U_{LC}^{LO}\} \quad (19)$$

1) Problem solving: Derivation-based optimization: In order to optimize two objectives of mode switching probability and utilization, the optimum value of n_i must be obtained for each task τ_i. If the uniform n is considered for all tasks to compute the $WCET_i^{LO}$, we can obtain the optimum n by finding the derivation of both objectives. Using the method of the second derivative helps to find the largest or smallest value of a function, where the derivative equals zero. Further details, on how the derivative works to find the optimum value, are provided in the result section (Section VI-B2) by an example. However, obtaining the uniform optimum n for all tasks is not fair and tasks have different time distribution. Table III shows the minimum value of n for some benchmarks of MiBench Suite, where $WCET_i^{opt} = ACET_i + n \times \sigma \geq WCET_i^{pes}$. Due to having different time distribution of tasks, choosing the uniform n causes the system's objectives to not optimize well and precisely. As a result, optimization techniques that can handle non-uniform values of n_i across different tasks and can scale effectively with increasing number of tasks in the system are necessary.

2) Problem solving: GA-based optimization: Global optimization methods based on randomized algorithms, have been used extensively in system-level design space exploration for QoS improvement in embedded systems [39]. In our current work, we use Genetic Algorithms (GA) for solving the maximization problem shown in Eq. (19). GA involves using randomized search methods based on the principles of natural evolution and genetics.

It is important to mention that Mixed-Integer Linear Programming (MILP) can be used as an alternative to Genetic Algorithms (GA) for optimisation. However, the problem formulation of MILP is much more complex compared to that of GA, which allows a simpler implementation of the fitness function. Although GA has the lack of optimality guarantees, MILP also does not scale very well with the number of integer variables. So, increased number of integer

TABLE IV
EXECUTION TIME DISTRIBUTION OF VARIOUS BENCHMARKS

Exe. Parameters (ms)	insertsort-10000	matrix-mult	qsort-10000	corner	edge	smooth	epic	bitcount	dijkstra	FFT
$WCET^{pes}$	753.23	387.67	759.32	51.63	131.47	301.09	230.81	1142.17	1039.98	686.52
$ACET$	51.33	13.05	39.65	0.55	0.94	9.317	2.69	64.73	81.95	6.15
σ	6.38	5.6	5.46	0.71	0.87	5.63	1.98	6.94	8.65	2.12

(and real) variables resulting from large number of tasks—n_i and support variables—in an MILP formulation can increase the complexity considerably. Most state-of-the-art tools for solving MILP problems also provide a time-bound *best-effort* solution for complex problems. Further, for the distribution-aware optimization for real-world tasks, we use a lookup table to search for the closest WCET and PMS values. Implementing such lookup-based optimizations from real-world observations with standard MILP formulation can be considerably more complex than using GA. It must be noted that the focus of the article is on showing the efficacy of the proposed methodology in providing improved trade-offs between mode switching probability and utilization. While we would ideally prefer optimization methods with guaranteed optimality, the choice of GA was based on the ease of implementation and the support for integrating varying estimation methods–both mathematical and lookup-based. However, MILP formulation for the current research problem can be a suitable topic for further exploration. The encoding approach and GA methods used in our current work include the following:

- *Individual*: An ordered sequence of integer values forms the individual in the population. Each integer in the sequence corresponds to the value of n_i for a task τ_i.
- *Population*: During the optimization we generate two types of individuals for initializing the population of the first generation of candidate solutions. Firstly, we generate individuals comprising of randomly sampled n_i values from the range $[1, 50]$ for each task τ_i in the benchmark. Secondly, we generate uniform-valued individuals from the same range to ensure that the optimization included uniform values of n_i for each task.
- *Cross-over* and *Mutation*: We used two-point cross-over for exchanging n_i values among two candidate solutions. During cross-over, the configurations of the two randomly selected possible solutions are interchanged. This process forms one of the algorithms that generates new possible solutions (individuals) for the next generation of solutions. In our current problem, this entails interchanging the n_i values of two candidate solutions, selected from the current generation, for a subset of the tasks. Similarly, we used single-point mutation to set the value of n_i for a randomly selected task in the candidate solution to a randomly selected value in the range $[1, 50]$.
- *Selection*: We use tournament selection for choosing the candidate solutions for the population of the next generation. It involves randomly choosing a fixed number of individuals from the current population and selecting the one with the maximum value of $(1 - P_{Sys}^{MS}) \times U_{LC}^{LO}$ for the next generation.
- *Fitness* and *Feasibility*: Eq. (16) to (18) were used to evaluate the fitness ($(1 - P_{Sys}^{MS}) \times U_{LC}^{LO}$) of each candidate solution. Similarly, Eq. (14) and (15) were used to determine the feasibility of each candidate solution.

VI. EVALUATION

In this section, we present the experiments to evaluate the effectiveness of our proposed scheme in terms of utilization, schedulability, and mode switching probability.

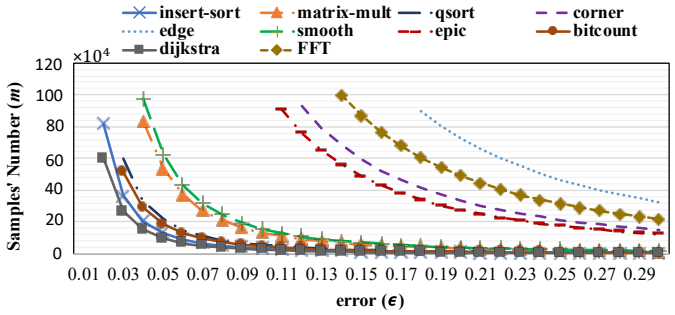
A. Evaluation with real-life benchmarks at run-time

*1) **Evaluation setup**:* To evaluate our scheme, we conducted some experiments on the ODROID XU4 board powered by ARM, which has the big.LITTLE architecture, with four Cortex A15 (big) and four Cortex A7 (LITTLE) cores. We use the LITTLE cores with the maximum frequency of $1.4GHz$, for doing the experiments.

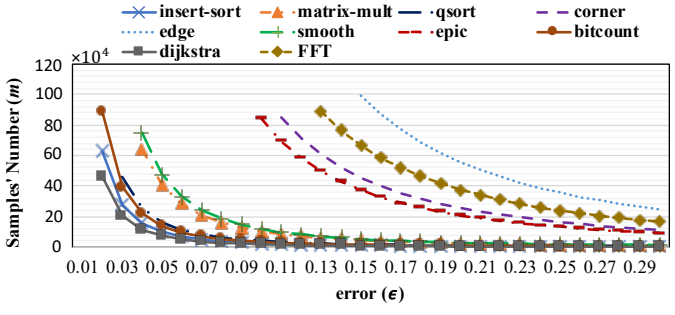
To evaluate our scheme by real benchmarks, we use various benchmarks from MiBench benchmark suite [36] such as automotive, network, telecomm. and from AXBench [40] such as matrix-multiplier. We execute each benchmark with different inputs on ODROID-XU4 board, to achieve their execution times. Table IV shows the pessimistic WCET, ACET, and σ (standard deviation) of these benchmarks.

*2) **Minimum required samples to estimate ACETs**:* Fig. 3 shows the minimum required samples of each benchmark based on *Theorem 2* to estimate ACET, by varying the parameters of (ϵ, δ)-*Approximation*. In fact, as an example in Fig. 3a, with 90% confidence, the estimation error of ACET for each benchmark is less than $(\epsilon \times \mu)$, where μ is the real mean. Besides, by decreasing the confidence, the minimum required samples for each benchmark is decreased. It means with more samples, we can say with more confidence that the difference between the estimated average and the real average is less than $\epsilon \times ACET^{real}$.

*3) **Investigating MC systems' timing behaviour**:* In order to evaluate the proposed approach, we run these benchmarks on a single core. We consider ‹insert-sort›, ‹matrix-mult›, ‹qsort›, ‹bitcount›, ‹dijkstra›, and ‹FFT› as HC tasks, and ‹corner›, ‹edge›, ‹smooth›, and ‹epic› as LC tasks. We compute the low WCET for each HC task based on the three polices– our scheme under *Chebyshev's theorem*, our scheme under distribution analysis, and the fraction analysis. In order to specify what the fraction analysis is, as discussed in Section II, most of the state-of-the-art approaches have defined a fraction of $WCET^{pes}$ as $WCET^{opt}$. For example, if we define $\lambda = \frac{WCET^{opt}}{WCET^{pes}}$, researchers in [12] have considered $\lambda \in [\frac{1}{2.5}, \frac{1}{1.5}]$ in their experiments. In [1], two different ranges for λ have been considered, $\lambda \in [\frac{1}{4}, 1]$ and $\lambda \in [\frac{1}{8}, 1]$. Researchers in [4] have considered the amount of $\lambda \in \{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\}$. Since all papers



(a)



(b)

Fig. 3. Required number of samples for different benchmarks by varying the error (ϵ) and confidence ($1-\delta$). (a) $1-\delta$= 0.9. (b) $1-\delta$= 0.8.

have the same policy to determine $WCET^{opt}$, we choose [1] as a representative of these approaches.

For these real tasks, including both LC and HC tasks, the system with $\lambda = 1$ has the utilization more than one in the worst-case scenario and then it is not schedulable. Therefore, we only consider the amount of λ as $\{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}\}$. Here, we investigate the system at run-time for 1000 hyper-period of tasks, and see how often these tasks exceed their $WCET^{opt}$ under various policies and the system has to switch to the HI mode.

By reducing the λ, the optimistic WCET (C_i^{LO}) for HC tasks decreases, and the system executes more LC tasks. But on the other hand, it causes frequent system switches and more LC tasks dropping during run-time, leading to lower QoS. As an example, Table V shows the maximum total utilization bound that can be assigned to LC tasks at design-time for each scenario and also the percentage of dropped LC tasks due to the system mode switching at run-time. We assume that the system needs to run different instances of each LC tasks (with different input) as much as possible to improve the QoS. As shown in Table V, the *Chebyshev*-based scheme can schedule more LC tasks in the system compare to [1] approach. Although the maximum assigned utilization to LC tasks is almost equal to the scenario of [1] with $\lambda = \frac{1}{8}$, the mode switching probability and the number LC dropped tasks are lower in the *Chebyshev*-based scheme. This is because a fraction of pessimistic WCET does not provide any information about how many samples might exceed it. So, setting the optimistic WCET of each task equal to $\lambda = \frac{1}{8}$ of the pessimistic WCET of that task is too low for some tasks and too high for others. The optimization goal in the last column of the table shows this fact that the *Chebyshev*-based

TABLE V
SYSTEM PERFORMANCE IN BOTH DESIGN-TIME AND RUN-TIME PHASES, FOR DIFFERENT SCENARIOS

	dropped LC jobs(%)	$max(U_{LC}^{LO})$	P_{Sys}^{MS}	$max(U_{LC}^{LO})\times(1-P_{Sys}^{MS})$
[1] $\lambda = \frac{1}{2}$	0	44.7%	0.24%	0.446
[1] $\lambda = \frac{1}{4}$	0	61.78%	1.21%	0.610
[1] $\lambda = \frac{1}{8}$	0.33%	76.37%	10.23%	0.686
[1] $\lambda = \frac{1}{16}$	39.29%	86.61%	92.02%	0.069
Chebyshev	0.06%	77.01%	7.1%	0.715
Dist. Analyt.	0	84.31%	2.25%	0.824

scheme performs better than the approach of [1] (i.e., the goal metric has a larger value). Besides, the distribution analytics-based scheme improves total utilization by 7.3% compared to *Chebyshev*-based scheme. It also reduces the mode switching probability by 4.85%. This is because the *Chebyshev* is a general formula that is valid for any distribution, but it isn't very optimistic. The value of optimization goal in the last column of the table also shows this fact. Let us consider the distribution analytics-based scheme with the method of [1] with $\lambda = \frac{1}{16}$ which both have almost the same total utilization. The results show that in the distribution analytics-based, the probability of mode switching and the percentage of dropped LC tasks are 89.75% and 39.29% lower, respectively, which is desirable.

B. Evaluation with synthetic task sets

1) ***Task set generation and evaluation setup:*** In order to further evaluate our scheme, we generated synthetic dual-criticality task sets similar to the state of the art studies [1], [13], [20], [21], for various system utilization bounds (U_{bound}) in line with the previous works [1], [13], [15], [20], [21], where ($U_{bound} = max(U_{LC}^{LO} + U_{HC}^{LO}, U_{HC}^{HI})$). The algorithm adds tasks to the task set randomly to increase the U_{bound} until it reaches a given threshold. We evaluate different approaches for U_{bound} in the range of [0.05, 1] with steps of 0.05 and for each U_{bound}, 1000 task sets are generated. Here, we consider balanced tasks in terms of criticality levels, i.e., the probability of a generated task being HC is equal to being LC. Besides, inspired by real execution times, presented in Table IV, we provide the $WCET^{pes}$, $ACET$ and σ in the range of [52,1142], [0.55,81.95] and [0.71,8.65]ms, respectively, where $WCET^{pes} > ACET$. As a result, the periods of tasks are computed based on the task utilization and $WCET^{pes}$ ($u_i^{HI} = \frac{WCET_i^{pes}}{P_i}$).

The recent advanced features in CAD tools, like MATLAB, Excel, and new libraries in Python, provide several practical ways to find a distribution that fits the best to the data samples. Besides, the probabilistic analysis for distribution fitting is implemented in Python using multiple packages, including scikit-learn. For solving the formulated problem with GA, we set the mutation probability to 0.2 and the cross-over probability to 0.8. We also used five individuals in the tournament selection process. The optimization methods were implemented in Python using the DEAP [41] package. In the following, we perform extensive simulations to evaluate the

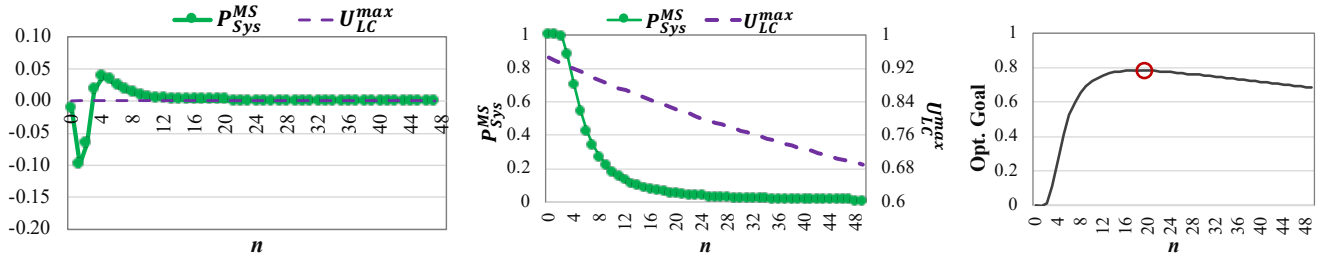


Fig. 4. Effect of varying uniform n on maximum assigned utilization to LC tasks and mode switching probability for an example task set. (a) second derivation of system properties. (b) P_{sys}^{MS} and $max(U_{LC}^{LO})$. (c) objective function.

effectiveness of our proposed approach in comparison with the state-of-the-art methods.

*2) **Effect of varying uniform n on maximum assigned utilization to LC tasks and mode switching probability for a task set example**:* In this section, we evaluate the effects of varying the parameter n, used to determine $WCET^{opt}$ for each HC task, on system properties. In this experiment, for the sake of presentation, we considered only one n (uniform) for all HC tasks. However, in further experiments, due to our explanation in Section V-E1, we find an independent n for each task with the help of the GA algorithm. As mentioned, we improve resource utilization by a significant utilization that can be allocated to LC tasks in the LO mode. Fig. 4 shows the results for an example task set with $U_{HC}^{HI} = 0.84$. First we show the results, solved by derivation-based optimization in Fig. 4a, and then by GA-based Optimization in Fig. 4b and 4c.

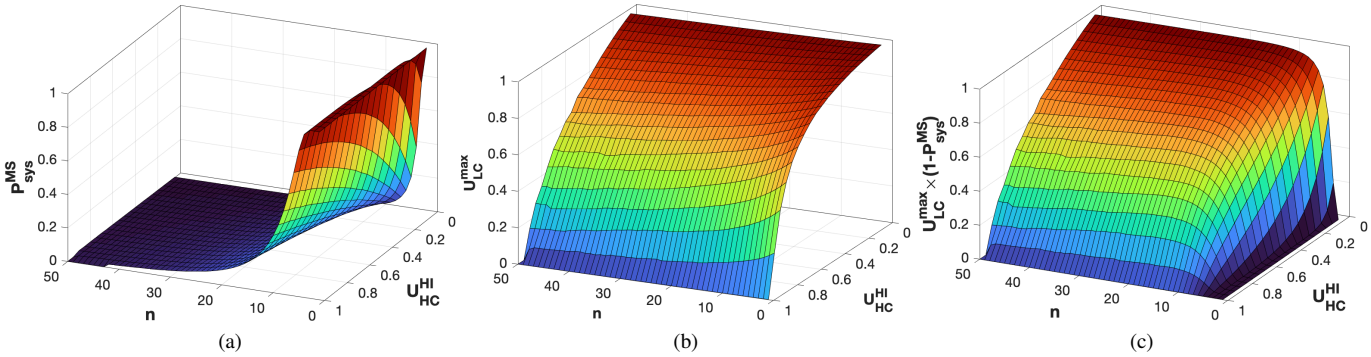
Fig. 4a shows the second derivative of utilization and mode switching probability for the task set. The figure shows that the second derivative of utilization is almost zero all the time, while the second derivative of mode switching probability becomes almost zero for $n \geq 18$. Therefore, we can conclude that the mode switching probability impacts more on obtaining the optimum value of n. To show how effective the derivation-based optimization is, we solve the problem with uniform n by GA-based Optimization for this example, shown in Fig. 4b and 4c.

Eq. (15) shows that by increasing the value of n, the $WCET^{opt}$ of HC tasks, and consequently HC tasks' utilization in the LO mode are increased, which reduces the number of scheduled LC tasks at design-time ($max(U_{LC}^{LO})$). On the other hand, Eq. (16) shows that by increasing the value of n, the probability of mode switching (P_{sys}^{MS}) is decreased, which means fewer LC tasks are dropped at run-time. Fig. 4b depicts that, by increasing the value of n, both P_{sys}^{MS} and $max(U_{LC}^{LO})$ are decreased, while to achieve the best utilization, we need to maximize $max(U_{LC}^{LO})$, and minimize P_{sys}^{MS}. Therefore, if the n is set to 5, then P_{sys}^{MS} is equal to 0.54, and $max(U_{LC}^{LO})$, is equal to 0.91. Meanwhile for $n = 10$, P_{sys}^{MS} is equal to 0.18, and $max(U_{LC}^{LO})$ is equal to 0.88. Indeed, P_{sys}^{MS} is decreased at a great rate by increasing n, compared to $max(U_{LC}^{LO})$ decrements. Now, consider $n = 20$ where $P_{sys}^{MS} = 0.05$ and $max(U_{LC}^{LO}) = 0.82$. It can be seen that the rate of P_{sys}^{MS} reduction is decreased by increasing n, while $max(U_{LC}^{LO})$ reduction rate is very low. Therefore, $max(U_{LC}^{LO})$ becomes more important than P_{sys}^{MS} in this case. We used Eq. (19), to find a proper n which makes a trade-off between P_{sys}^{MS} and $max(U_{LC}^{LO})$ and improves the system utilization. Fig. 4c shows, the optimum n is 18 for our case study task set where $max(U_{LC}^{LO}) = 83\%$ and $P_{sys}^{MS} = 0.06$.

*3) **Effect of varying uniform n on maximum assigned utilization to LC tasks and mode switching probability for more task sets**:* Now, we evaluate the effects of parameter n and different utilization of HC tasks on system properties in Fig. 5, by running 1000 task sets for each utilization point. According to Fig. 5a, P_{sys}^{MS} is increased when utilization increases. For example, for a constant $n = 10$, for U_{HC}^{HI} equal to 0.4 and 0.8, P_{sys}^{MS} is 15.47% and 28.43%, respectively. The reason is, when utilization of HC tasks is high, more HC tasks are scheduled in the system. Since each HC task has the probability of overrunning, by increasing the number of HC tasks, P_{sys}^{MS} is increased. In addition, we discussed that P_{sys}^{MS} is decreased by increasing n. Fig. 5b also shows that by increasing U_{HC}^{HI}, there is less opportunity to schedule LC tasks. As a result, the system schedules fewer LC tasks which degrades $max(U_{LC}^{LO})$. As an example, for a constant $n = 10$, if $U_{HC}^{HI} = 0.4$, then $max(U_{LC}^{LO}) = 87.59\%$ and if $U_{HC}^{HI} = 0.8$, then $max(U_{LC}^{LO}) = 53.46\%$. Besides, as mentioned, increasing n causes a decrease in $max(U_{LC}^{LO})$. As a result, by increasing n, P_{sys}^{MS} is reduced (which is desirable), and the LC tasks utilization and consequently schedulability is also reduced (which is not desirable). Now, if we optimize both P_{sys}^{MS} and assigned utilization to LC tasks, we can find the optimum value of n for HC tasks. Fig. 5c shows the product of P_{sys}^{MS} and $max(U_{LC}^{LO})$ (Eq. 19), where, the optimum n is decreased in general with an increase in U_{HC}^{HI}, to run more tasks in system.

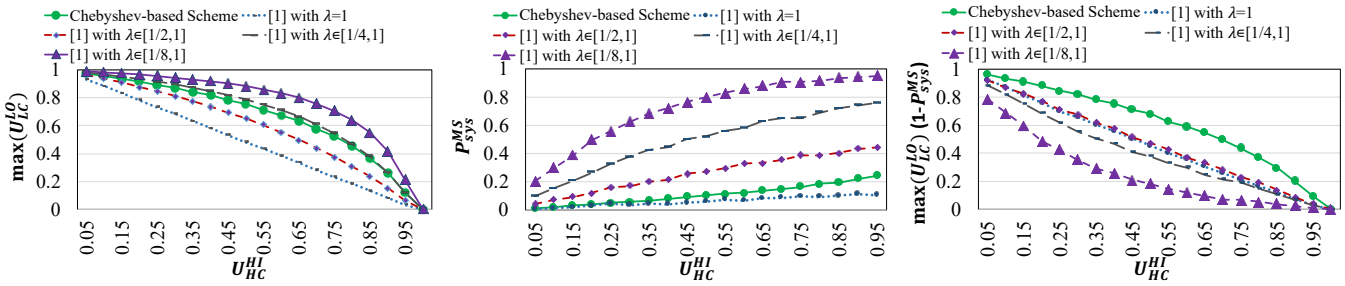
*4) **Comparison with the other policies**:* Since applications have different time distributions, choosing the uniform n prevents the system from optimizing its objectives precisely. Therefore, solving the problem with optimization algorithms like GA is the best method to optimize system properties. As a result, in this subsection, we compare the mode switching probability and resource utilization under our proposed scheme with non-uniform n using the GA-algorithm, and the other policies, used to determine $WCET^{opt}$ and then, U_{HC}^{LO}.

Since ACET and σ for each task are known, the system mode switching probability for other policies can be obtained using Eq. 6. Fig. 6 shows the results of comparing different



(a) (b) (c)

Fig. 5. Effect of n and HC tasks' utilization on maximum assigned utilization to LC tasks and mode switching probability. (a) P_{sys}^{MS} by varying n and U_{HC}^{HI}. (b) $max(U_{LC}^{LO})$ by varying n and U_{HC}^{HI}. (c) optimization goal.



(a) (b) (c)

Fig. 6. The effectiveness of our proposed scheme in comparison with other policies, proposed in other research works. (a) $max(U_{LC}^{LO})$ by varying U_{HC}^{HI}. (b) P_{sys}^{MS} by varying U_{HC}^{HI}. (c) optimization goal by varying U_{HC}^{HI}.

policies and our scheme with the optimum n_i for each task τ_i of task sets using the GA-algorithm, for different utilization. In the Baruah's approach [1], considering a large lower-bound value for λ like 1, $\frac{1}{2}$, reduces the probability of mode switching, but it under-utilizes the system during run-time. For example if $U_{HC}^{HI} = 0.75$, for $\lambda = 1$, $P_{sys}^{MS} = 9.66\%$ and $max(U_{LC}^{LO}) = 23.28\%$, while for our proposed scheme, $P_{sys}^{MS} = 16.46\%$ and $max(U_{LC}^{LO}) = 52.39\%$. On the other hand, using a smaller lower-bound value for λ like $\frac{1}{8}$, increases the maximum utilization of the LC tasks with high mode switching probability. For instance, if $U_{HC}^{HI} = 0.75$, then $P_{sys}^{MS} = 90.75\%$ and $max(U_{LC}^{LO}) = 70.75\%$. Note that, to prevent the figures from being unclear, we only show the result for the $\lambda \in [\frac{1}{8}, 1]$. The results for $\lambda \in [\frac{1}{16}, 1]$ and $\lambda \in [\frac{1}{32}, 1]$, have more maximum utilization increment of the LC tasks with higher mode switching probability in comparison with $\lambda \in [\frac{1}{8}, 1]$, which is undesirable. Our approach works well in both system properties by determining the best $WCET^{opt}$ values for HC tasks base on the ACET, and then, the optimum U_{HC}^{LO}. Fig. 6c shows this fact by optimizing both system properties, where the proposed scheme performs better than other policies. As a result, our scheme improves the utilization by up to 72.27% compared to the existing approaches, while P_{sys}^{MS} is bounded by 24.28% in the worst-case scenario.

*5) **Evaluating scheduling approaches under proposed scheme**:* Now, we evaluate and compare the results in terms of schedulable task sets (acceptance ratio) to the state-of-the-art approaches, proposed in [1], [2], with and without our scheme. In this experiment, we assume that the probability that a task is an HC or LC is equal. In both [1], [2], the EDF-VD algorithm has been used to schedule the tasks. In [2], the algorithm executes all LC tasks in the HI mode by reducing their WCET to 50%, and also in [1], the algorithm drops all LC tasks when the system switches to the HI mode. It is noteworthy to mention that our scheme for selecting the suitable $WCET^{opt}$ for HC tasks can be applied to any scheduling algorithm with any policy of task execution and optimize the resource utilization and mode switching probability.

Fig. 7 shows the acceptance ratio for two state-of-the-art scheduling approaches [1], [2], which are improved with our scheme in all utilization bounds. As shown in this figure, when $U_{bound} \leq 0.7$, all tasks sets are schedulable with Liu's approach [2], and our scheme. When the system utilization is increased ($0.7 < U_{bound} \leq 0.95$), our proposed scheme performs better than Liu's approach [2] in terms of acceptance ratio. And so that, no task set is schedulable for $U_{bound} \geq 0.95$. Besides, the same trend is found for Baruah's approach [1]. The reason for having a better acceptance ratio in our scheme is determining the appropriate $WCET^{opt}$ for HC tasks and executing more tasks in the system.

VII. Conclusion

In this paper, we proposed a scheme, Worst-Case Execution Time (WCET) analysis of mixed-criticality tasks, which manages the probability of mode switching and improves the

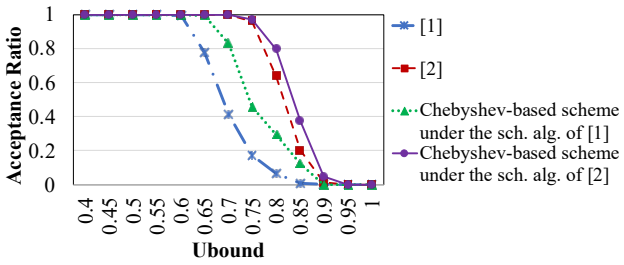


Fig. 7. Different scheduling approaches with our scheme

schedulability and timing budget allocated to low-criticality tasks. Our scheme analyzes the application in the offline phase to determine an appropriate low WCET for each task, based on two approaches. In the first approach, we analyze the applications based on the *Chebyshev theorem*, a general theorem which is valid for any task with any execution time distribution. However, to have a tighter bound for system mode switching probability, we analyze the applications based on their distribution. The proposed scheme based on the *Chebyshev theorem* improves the system utilization and schedulability up to 72.27% and 91.2%, respectively, while bounding the mode switching probability to 24.28% in the worst-case scenario. We also evaluated the approaches with real benchmarks on a hardware platform to show its efficiency. The proposed scheme based on the distribution analysis can reduce the mode switching probability 4.85% more for a real task set, compared to the scheme based on the *Chebyshev theorem*.

As future work, we intend to work on a machine-learning-based execution time bound for the application, which would determine better WCETs for HC tasks based on the run-time behaviour. Therefore, we would present a scheduling algorithm and execute the lower-criticality tasks in higher mode.

REFERENCES

[1] 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. of Euromicro Conference on Real-Time Systems (RTNS)*, 2012, pp. 145–154.

[2] 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. on IEEE Real-Time Systems Symposium (RTSS)*, 2016, pp. 35–46.

[3] A. Burns and R. I. Davis, "A Survey of Research into Mixed Criticality Systems," *ACM Comput. Surv.*, vol. 50, no. 6, 2017.

[4] 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. on IEEE Real-Time Systems Symposium (RTSS)*, 2018, pp. 373–383.

[5] 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 *2019 22nd Euromicro Conference on Digital System Design (DSD)*, 2019, pp. 546–553.

[6] S. S. Sahoo, B. Ranjbar, and A. Kumar, "Reliability-aware resource management in multi-/many-core systems: A perspective paper," *Journal of Low Power Electronics and Applications*, vol. 11, no. 1, p. 7, 2021.

[7] L. A. Johnson, "DO-178B, Software considerations in airborne systems and equipment certification," *Crosstalk, October*, vol. 199, 1998.

[8] B. Ranjbar, B. Safaei, A. Ejlali, and A. Kumar, "FANTOM: Fault Tolerant Task-Drop Aware Scheduling for Mixed-Criticality Systems," *IEEE Access*, vol. 8, pp. 187 232–187 248, 2020.

[9] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The Worst-Case Execution-Time Problem—Overview of Methods and Survey of Tools," *ACM Trans. Embed. Comput. Syst. (TECS)*, vol. 7, no. 3, May 2008.

[10] A. Kumar, B. Mesman, H. Corporaal, and Y. Ha, "Iterative Probabilistic Performance Prediction for Multi-Application Multiprocessor Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, no. 4, pp. 538–551, 2010.

[11] C. Ballabriga, H. Cassé, C. Rochange, and P. Sainrat, "OTAWA: an open toolbox for adaptive WCET analysis," in *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*. Springer, 2010, pp. 35–46.

[12] 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 Transactions on Computers (TC)*, vol. 67, no. 7, pp. 975–991, 2018.

[13] 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 Transactions on Computers (TC)*, vol. 67, no. 4, pp. 543–558, 2018.

[14] B. Ranjbar, T. D. A. Nguyen, A. Ejlali, and A. Kumar, "Power-Aware Run-Time Scheduler for Mixed-Criticality Systems on Multi-Core Platform," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 40, no. 10, pp. 2009–2023, 2021.

[15] C. Gu, N. Guan, Q. Deng, and W. Yi, "Partitioned mixed-criticality scheduling on multiprocessor platforms," in *Proc. on Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–6.

[16] B. Ranjbar, A. Hosseinghorban, S.S. Sahoo, A. Ejlali, and A. Kumar, "Improving the Timing Behaviour of Mixed-Criticality Systems Using Chebyshev's Theorem," in *Proc. on Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 264–269.

[17] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[18] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. on IEEE Real-Time Systems Symposium (RTSS)*, 2007, pp. 239–243.

[19] S. Baruah and S. Vestal, "Schedulability analysis of sporadic tasks with multiple criticality specifications," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 2008, pp. 147–155.

[20] Z. Al-bayati, J. Caplan, B. H. Meyer, and H. Zeng, "A four-mode model for efficient fault-tolerant mixed-criticality systems," in *Proc. on Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 97–102.

[21] Z. Guo, L. Santinelli, and K. Yang, "EDF Schedulability Analysis on Mixed-Criticality Systems with Permitted Failure Probability," in *Proc. of the International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2015, pp. 187–196.

[22] 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, May 2015.

[23] X. Gu and A. Easwaran, "Dynamic Budget Management with Service Guarantees for Mixed-Criticality Systems," in *Proc. on IEEE Real-Time Systems Symposium (RTSS)*, 2016, pp. 47–56.

[24] X. Gu and A. Easwaran, "Dynamic budget management and budget reclamation for mixed-criticality systems," *Real-Time Systems*, vol. 55, no. 3, pp. 552–597, 2019.

[25] L. Santinelli and L. George, "Probabilities and mixed-criticalities: the probabilistic c-space," in *Proc. on IEEE Real-Time Systems Symposium (RTSS)*, 2015.

[26] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran, "Probabilistic Analysis for Mixed Criticality Systems Using Fixed Priority Preemptive Scheduling," in *Proc. of Euromicro Conference on Real-Time Systems (RTNS)*, 2017, p. 237–246.

[27] L. De Haan and A. Ferreira, *Extreme value theory: an introduction*. Springer Science & Business Media, 2007.

[28] S. Jiménez Gil, I. Bate, G. Lima, L. Santinelli, A. Gogonel, and L. Cucu-Grosjean, "Open Challenges for Probabilistic Measurement-Based Worst-Case Execution Time," *IEEE Embedded Systems Letters*, vol. 9, no. 3, pp. 69–72, 2017.

[29] F. Reghenzani, L. Santinelli, and W. Fornaciari, "Dealing with Uncertainty in PWCET Estimations," *ACM Trans. Embed. Comput. Syst. (TECS)*, vol. 19, no. 5, Sep. 2020.

[30] A. Bhuiyan, F. Reghenzani, W. Fornaciari, and Z. Guo, "Optimizing energy in non-preemptive mixed-criticality scheduling by exploiting

probabilistic information," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 11, pp. 3906–3917, 2020.
[31] B. Hu, L. Thiele, P. Huang, K. Huang, C. Griesbeck, and A. Knoll, "FFOB: Efficient online mode-switch procrastination in mixed-criticality systems," *Real-Time Systems*, vol. 55, no. 3, pp. 471–513, 2019.
[32] B. Ranjbar, A. Hosseinghorban, M. Salehi, A. Ejlali, and A. Kumar, "Toward the design of fault-tolerance-and peak-power-aware multi-core mixed-criticality systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, pp. 1–1, 2021.
[33] C. Therrien and M. Tummala, *Probability and random processes for electrical and computer engineers*. CRC press, pp. 190, 2018.
[34] A. Hoseinghorban, M. R. Bahrami, A. Ejlali, and M. A. Abam, "Chance: Capacitor charging management scheme in energy harvesting systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 40, no. 3, pp. 419–429, 2021.
[35] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The collected works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.
[36] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Proc. of IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, 2001, pp. 3–14.
[37] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.
[38] E. B. Holmgren, "The pp plot as a method for comparing treatment effects," *Journal of the American Statistical Association*, vol. 90, no. 429, pp. 360–365, 1995.
[39] S. S. Sahoo, B. Veeravalli, and A. Kumar, "Cl(r)early: An early-stage dse methodology for cross-layer reliability-aware heterogeneous embedded systems," in *Proc. on Design Automation Conference (DAC)*, 2020, pp. 1–6.
[40] A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, and P. Lotfi-Kamran, "Axbench: A multiplatform benchmark suite for approximate computing," *IEEE Design & Test*, vol. 34, no. 2, pp. 60–68, 2017.
[41] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, no. 1, p. 2171–2175, Jul. 2012.

Behnaz Ranjbar received the B.S. degree in computer engineering from Amirkabir University of Technology in 2012 and the M.S. degree from Sharif University of Technology, Tehran, Iran in 2014. She is currently a joint Ph.D. student with the Sharif University of Technology, Tehran, Iran, and the Chair for Processor Design, Technische Universität Dresden, Dresden, Germany. Her research interest includes real-time, fault tolerant and low-power embedded system design.

Ali Hosseinghorban received his B.Sc. in computer engineering from Shahid Beheshti University and his M.Sc. from Sharif University of Technology in 2015 and 2017, respectively. He is currently a Ph.D. student in the Computer Engineering department of Sharif University of Technology, Tehran, Iran. He is also a visiting researcher at the Chair for Processor Design, CFAED, Technische Universität Dresden, Dresden, Germany. His research interests include low-power real-time embedded systems, energy harvesting systems and emerging nonvolatile memories.

Siva Satyendra Sahoo is currently working as a Postdoctoral Researcher with the Chair for Processor Design at TU Dresden. He received his doctoral degree (Ph.D., 2015-2019) in the field of reliability in heterogeneous embedded systems from the National University of Singapore, Singapore. He completed his masters (M.Tech, 2010-2012) from the Indian Institute of Science, Bangalore in the specialization Electronics Design Technology. He has also worked with Intel India, Bangalore in the domain of Physical Design. His research interests include Embedded Systems, Machine Learning, Approximate Computing, Reconfigurable Computing, Reliability-aware Computing Systems, and System-level Design.

Alireza Ejlali received the PhD degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2006. He is currently an associate professor of computer engineering at SUT. From 2005 to 2006, he was a visiting researcher in the Electronic Systems Design Group, University of Southampton, Southampton, United Kingdom. He is currently the Director of the Embedded Systems Research Laboratory, Department of Computer Engineering, Sharif University of Technology. His research interests include low power design, real-time, and fault-tolerant embedded systems.

Akash Kumar (SM'13) 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 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.