Improving the Timing Behaviour of Mixed-Criticality Systems Using Chebyshev's Theorem

Behnaz Ranjbar*†, Ali Hoseinghorban†, Siva Satyendra Sahoo*, Alireza Ejlali†, and Akash Kumar*
*Chair for Processor Design, CFAED, Technische Universität Dresden, Dresden, Germany
†Embedded System Research Laboratory (ESRLab), Sharif University of Technology, Tehran, Iran
{behnaz.ranjbar, siva_satyendra.sahoo, akash.kumar}@tu-dresden.de, hoseinghorban@ce.sharif.edu, ejlali@sharif.edu

***Abstract*—In Mixed-Criticality (MC) systems, there are often multiple Worst-Case Execution Times (WCETs) for the same task, corresponding to system operation mode. Determining the appropriate WCETs for lower criticality modes is non-trivial; while on the one hand, a low WCET for a mode can improve the processor utilization in that mode, on the other hand, using a larger WCET ensures that the mode switches are minimized, thereby maximizing the quality-of-service for all tasks, albeit at the cost of processor utilization. Although there are many studies to determine WCET in the highest criticality mode, no analytical solutions are proposed to determine WCETs in other 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 runtime as a result of frequent mode switches. Our experimental results show that our scheme improves the utilization of state-of-the-art MC systems by up to 85.29%, while maintaining 9.11% mode switching probability in the worst-case scenario.**

***Index Terms*—Mixed-Criticality, Mode Switching Probability, Resource Utilization, Schedulability, WCETs' Analysis.**

I. Introduction

NOWADAYS, Mixed-Criticality (MC) systems are widely used in embedded real-time systems to meet cost, space, timing, and power consumption requirements [1]–[4]. A wide range of safety-critical applications found in medical devices, automotive, and avionics industries are evolving into MC systems. These systems consist of tasks with multiple criticality levels. At runtime, the system must guarantee the successful execution of all high-criticality (HC) tasks to prevent catastrophic damages while ensuring that the processor utilization and Quality-of-Service (QoS) are maximized (execute a higher number of low-criticality (LC) tasks) [3]–[5].

In conventional real-time systems, the tasks are scheduled based on their pessimistic Worst-Case Execution Time (WCET). Many approaches like what presented in [6], [7] and tools like OTAWA [8] are used to determine the pessimistic WCET of a task by analyzing the task's control flow. These tools provide a safe and conservative execution time-bound so that no task's execution time exceeds it under any circumstances. However, Fig. 1 [6] depicts that most samples' execution time is significantly shorter than such a conservative WCET, which leads to poor processor utilization and QoS in conventional real-time systems.

To this end, in MC systems, there are usually multiple WCETs, corresponding to the multiple criticality levels and the on-going mode of operation [2], [9]–[11]. This ensures 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). MC systems, in the beginning, execute tasks based on the schedule with optimistic WCETs at runtime. If the execution time of at least one HC task exceeds its optimistic WCET, the system switches to the HI mode. In this mode, the pessimistic WCETs of HC tasks are considered to guarantee the correct execution of HC tasks, and the system has to drop all or some LC tasks [1]–[4], [11], [12]. Therefore, when the gap between the optimistic and pessimistic WCETs is large, more tasks, especially LC tasks, are scheduled at design-time. However, it may cause frequent system mode switches, and consequently, drop more LC tasks at runtime. When this gap is small, it reduces processor utilization due to the scheduling of fewer tasks at design-time.

This work is supported in part by the German Research Foundation (DFG) within the Cluster of Excellence Center for Advancing Electronics Dresden (CFAED) at the Technische Universität Dresden.

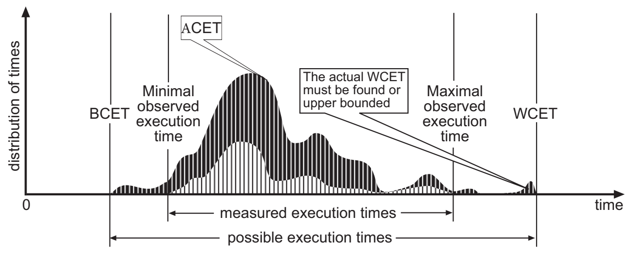


Fig. 1: Execution time distribution for a real-time task [6]. The figure shows the large gap between the WCET and the ACET.

Optimistic WCETs play an important role in designing the MC systems. Most state-of-the-art research works [2], [4], [9], [10], [12] 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). Furthermore, most studies have not analyzed the probability of exceeding the optimistic WCETs in system design.

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 LC tasks that can be guaranteed to meet the deadlines at design-time and the probability of mode switching at runtime to improve the system utilization and QoS. This paper proposes a novel scheme for MC systems to determine the appropriate optimistic WCETs for tasks. We focused on MC systems with two criticality levels, but our scheme could be used for MC systems with several criticality levels.

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.

- Formulating and solving (using Genetic Algorithm (GA)) an optimization problem for improving the resource utilization and reducing the mode switching probability.
- Evaluating our proposed scheme for various state-of-the-art MC systems to investigate their timing behaviour and system schedulability.

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

II. Related Works

A significant number of papers have been published in the last decade about designing the MC systems. Vestal [13], presented the MC task model and introduced different WCET levels for the first time. 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. Most of the approaches, such as [2], [4], [9], [10], [12], [14], generally set 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. 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 [15], [16] have determined the $WCET^{opt}$ of tasks at runtime, 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 [17], [18], have focused on probability distributions in MC systems by exploiting Extreme Value Theory (EVT) for timing analysis. Applying these estimation methods causes 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 [19]–[21]. Researchers in [21] 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 operational model definition for running the LC tasks is different from the popular MC model.

Some research works such as [22] have addressed mode switching probability in MC systems and how to have the safe mode switching at runtime. However, the relation between the HC tasks' $WCET^{opt}$ and mode switching probability has not been discussed.

Therefore, an appropriate WCETs 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 [1], [9], [21], in which multiple periodic tasks with two criticality levels are executed upon the same platform. Each system has a finite number of MC tasks, $\{\tau_1, \tau_2, ..., \tau_t\}$. In addition, DO-178B [23] is an industrial standard that is used in the avionic industry and has introduced 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), 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. Each task has one of these criticality levels. 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 [1].
- 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], [9].

Although the proposed scheme is independent of the precedence relationship, we consider an independent periodic task model to analyze the schedulability. 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 $u_i^l = \frac{C_i^l}{P_i}$ where $l \in \{LO, HI\}$.

System Operational Model: At first, the system starts 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 ready HC task in the processor's queue.

IV. Proposed Scheme

In this section, at first, a motivational example is presented in Section IV-A. Then, we propose our scheme for determining the optimistic WCETs. In the end, we discuss about the scheduling policy and optimization problem in Section IV-C.

A. Motivational example

In this example, we executed 20000 instances of five applications, and the ACET and WCET of them in terms of cpu clock cycle are presented in TABLE I. $WCET^{pes}$ of each application is determined by OTAWA [8]. 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], [12]. The important point that the table shows, is by increasing the input size of 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 $WCET^{pes}$

[1] Since we only consider two criticality levels in the paper, ($\zeta_i \in [LC, HC]$)

TABLE I: Comparison between ACET and WCET of different applications

Application	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

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%. So, it is not feasible to determine a general formula to find $WCET^{opt}$ based on $WCET^{pes}$. On the other hand, the mode switching probability behavior is more predictable when the $WCET^{opt}$ is set to ACET. However, simply setting $WCET^{opt}$ equal to ACET leads to many system mode changes (almost half of the instances).

In this paper, we introduce a scheme that provides a general formula to choose a suitable $WCET^{opt}$ based on ACET to make 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 $WCET^{opt}$ to improve the utilization of the system.

B. Determining optimistic WCET and overrunning probability

As mentioned earlier, determining the appropriate $WCET^{opt}$ for HC tasks is a major problem when designing the MC systems. To address this challenge, the proposed scheme designs the MC systems and analyzes the MC tasks of the application in the offline phase, and chooses a suitable $WCET^{opt}$ for each HC task based on their ACET, which improves the task schedulability and executes more LC tasks due to the big gap between the ACET and WCET. To determine $WCET^{opt}$, we introduce the following theorem based on the Chebyshev's theorem.

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

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

***One-Sided Chebyshev** [24]*: 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 $ACET_i$, the standard deviation execution time σ_i of 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 execute more 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 each execution time level for any task, independent of its distribution. To determine $WCET^{opt}$, *Chebyshev's theorem* can be applied, which requires mean $(ACET)$ and standard deviation of the execution time (σ) of each task. Thus, we compute $WCET^{opt}$ of task τ_i, as shown in Eq. 6, by determining the efficient value of n, while the probability of exceeding it, is $P_i^{MS} = \frac{1}{1+n^2}$. Hence, our proposed method does not determine the WCET at the HI mode $(WCET^{pes})$, which has been studied in many works [6].

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

In Section V, we evaluate the impact of different values of n in computing the $WCET^{opt}$ and the probability of system mode switching (P_{sys}^{MS}).

C. Schedulability analysis and optimization problem

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 such as [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:

$$U_{HC}^{LO} = \sum_{\zeta_i = HC} \frac{ACET_i + n_i \times \sigma_i}{T_i} \quad and \quad U_{HC}^{HI} = \sum_{\zeta_i = HC} \frac{C_i^{HI}}{T_i} \tag{7}$$

To choose a suitable $WCET^{opt}$ for each HC task, the optimum n_i (used in Eq. 6) for each HC task τ_i must be determined that minimizes the mode switching probability and maximizes the resource utilization by assigning more utilization to LC tasks, while all HC tasks are scheduled. To solve this, we formulate the optimization problem to find the optimum n_i for each task τ_i and determine its $WCET_i^{opt}$. From the perspective of task schedulability, Eq. 8 must be satisfied to guarantee schedulability by EDF-VD at runtime [1]. This equation presents the necessary and sufficient conditions to guarantee the task schedulability in both LO and HI modes, and deadline meeting of running tasks while the system switches to the HI mode [1].

$$U_{HC}^{LO} + U_{LC}^{LO} \leq 1 \quad and \quad U_{HC}^{HI} + \frac{U_{HC}^{LO} \times U_{LC}^{LO}}{1 - U_{LC}^{LO}} \leq 1 \tag{8}$$

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

$$ACET_i + n_i \times \sigma_i \leq WCET_i^{pes} \tag{9}$$

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 mode switching probability minimization. Let P_{Sys}^{MS} denotes the probability of system mode switching. If P_{Sys}^{noMS} is the probability that no HC task overruns and consequently, no mode switches happens, therefore, $P_{Sys}^{MS} = 1 - P_{Sys}^{noMS}$. Since tasks are independent, P_{Sys}^{MS} is computed as shown in Eq. 10, where P_i^{MS} is the probability of task overrunning for task τ_i. According to our discussion in Section IV-B, P_i^{MS}= $\frac{1}{1+n_i^2}$. The higher the n_i, the less the mode switching probability.

$$P_{Sys}^{MS} = 1 - \prod_{\zeta_i \in HC} (1 - P_i^{MS}) = 1 - \prod_{\zeta_i \in HC} (1 - \frac{1}{1 + n_i^2}) \tag{10}$$

***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, but it is upper-bounded by the schedulability constraints, which are written in Eq. 8. In this equation, 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.

$$U_{LC}^{LO} \leq 1 - U_{HC}^{LO} = 1 - \sum_{\zeta_i = HC} \frac{ACET_i + n_i \times \sigma_i}{T_i} \tag{11}$$

$$U_{LC}^{LO} \leq \frac{1 - U_{HC}^{HI}}{1 - U_{HC}^{HI} + U_{HC}^{LO}} = \frac{1 - U_{HC}^{HI}}{1 - U_{HC}^{HI} + \sum_{\zeta_i = HC} \frac{ACET_i + n_i \times \sigma_i}{T_i}} \tag{12}$$

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.

$$maximize\{(1 - P_{Sys}^{MS}) \times U_{LC}^{LO}\} \tag{13}$$

Problem Solving: We used Genetic Algorithms, a commonly used randomized search method, to solve the optimization problem (Eq. 13). The *population* constituted a collection of *individuals* made of randomly sampled n_i values for each task τ_i, in the benchmark. Two-point *cross-over* and single-point *mutation* were used to generate the feasible individuals for the next *generation*. Tournament *selection* was used to select the final set of individuals of the new generation.

V. EXPERIMENTS

In this section, we present the experiments to evaluate the effectiveness of our proposed scheme in terms of utilization, schedulability and mode switching probability. We evaluate our scheme by real benchmarks, and synthetic dual-criticality task sets similar to the state of the art studies [1], [10], [14]. The synthetic task sets are generated for various system utilization bounds (U_{bound}) in line with the previous works [1], [10], [12], [14]. 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. For each U_{bound}, 1000 task sets are generated, and the periods of tasks are selected in the range of [100, 900]ms.

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 [25] package.

A. Analyzing the mode switching probability

First, we evaluate the mode switching probability, and TABLE II shows the percentage of overruns for both analysis and experiments. In the analysis, we proved that the probability of overrunning for task τ_i with $WCET_i^{opt}$ ($WCET_i^{opt}$=$ACET_i + n_i \times \sigma_i$), is less than $P_i^{MS} = \frac{1}{1+n_i^2}$. In the experiments, we execute five applications with 20000 different inputs with MEET [26], an arm-based simulator, to achieve their execution times. TABLE II shows that the P_i^{MS} for each task in the experiments is much less than the analysis. These results were expected because the analysis provides an upper bound which is valid for any task with any execution time distribution.

B. Effect of varying uniform n on maximum assigned utilization to LC tasks and mode switching probability

We further 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, 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. 2 shows the results for an example task set with $U_{HC}^{HI} = 0.85$. Eq. 9 shows that by increasing the value of n, the $WCET^{opt}$ of HC tasks,

TABLE II: The effect of n on task overrunning

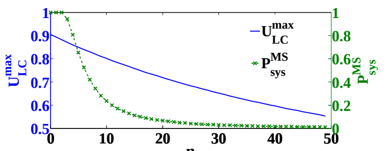
	Analysis	qsort-100	corner	edge	smooth	epic
n=0	100%	50.22%	53.27%	54.88%	54.31%	52.85%
n=1	50%	15.78%	9.88%	9.8%	9.40%	8.32%
n=2	20%	2.36%	3.07%	3.07%	2.97%	3.02%
n=3	10%	0.22%	0.01%	0.01%	0.06%	0.0%
n=4	5.88%	0.02%	0.0%	0.0%	0.0%	0.0%

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. 10 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 runtime. Fig. 2a 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}. As shown, if $n = 5$, then $P_{sys}^{MS} = 0.65$ and $max(U_{LC}^{LO}) = 0.84$ and for $n = 10$, $P_{sys}^{MS} = 0.24$ and $max(U_{LC}^{LO}) = 0.80$. 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 $P_{sys}^{MS} = 0.07$ and $max(U_{LC}^{LO}) = 0.71$. It can be seen that the rate of P_{sys}^{MS} reduction is decreased by increasing n, while the rate of $max(U_{LC}^{LO})$ reduction is less. Therefore, $max(U_{LC}^{LO})$ becomes more important than P_{sys}^{MS} in this case. We used Eq. 13, to find a proper n which make a trade-off between P_{sys}^{MS} and $max(U_{LC}^{LO})$ and improve the system utilization. Fig. 2b shows, the optimum n is 18 for our case study task set that $max(U_{LC}^{LO}) = 73\%$ and $P_{sys}^{MS} = 0.08$.

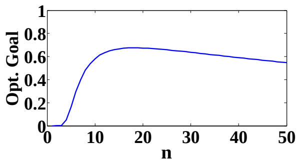
Now, we evaluate the effects of parameter n and different utilization of HC tasks on system properties in Fig. 3, by running 1000 task sets for each utilization point. According to Fig. 3a, P_{sys}^{MS} is increased when there is an increase in utilization. For example, for a constant $n = 10$, P_{sys}^{MS} in $U_{HC}^{HI} = 0.4$ and 0.8, is 12.91% and 23.71%, 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 with an increase in n. Fig. 3b also shows that, by increasing U_{HC}^{HI}, referring to the scheduling algorithm and schedulability conditions (Eq. 11 and 12), less core utilization is assigned to LC tasks. As an example for a constant $n = 10$, if $U_{HC}^{HI} = 0.4$, then $max(U_{LC}^{LO}) = 97.94\%$ and if $U_{HC}^{HI} = 0.8$, then $max(U_{LC}^{LO}) = 88.17\%$. 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 satisfactory), 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. 3c shows the product of P_{sys}^{MS} and $max(U_{LC}^{LO})$ (Eq. 13), where, the optimum n is decreased in general with an increase in U_{HC}^{HI}, to run more tasks in system.

C. Comparison with the other policies

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}. As discussed in Section II, most of the state-of-the-art approaches



(a) P_{sys}^{MS} and $max(U_{LC}^{LO})$



(b) objective function

Fig. 2: Effect of varying uniform n on maximum assigned utilization to LC tasks and mode switching probability for an example task set with $U = 0.45$

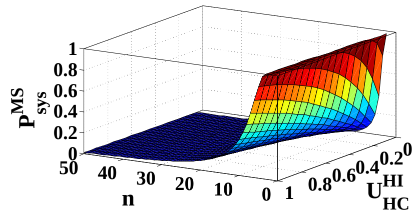
have defined a fraction of $WCET^{pes}$ as $WCET^{opt}$. For example, if we define $\lambda = \frac{WCET^{opt}}{WCET^{pes}}$, researchers in [9] have considered $\lambda \in [\frac{1}{2.5}, \frac{1}{1.5}]$ in their experiments. In [1], two different ranges for λ have been considered, $\lambda \in [\frac{1}{4}, 1]$ and $\lambda \in [\frac{1}{8}, 1]$. Researchers in [4] have considered the amount of $\lambda = \{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\}$. Since all papers have the same policy in determining $WCET^{opt}$, we choose [1] as a representative methodology of all research works in the experiments.

Since ACET and σ for each task are known, the system mode switching probability for other policies can be obtained using Eq. 6. Fig. 4 and 5 show the results of comparing different 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 $\frac{1}{8}, \frac{1}{4}$, reduces the probability of mode switching, but it under-utilizes the system during runtime. For example if $U_{HC}^{HI} = 0.8$, for $\lambda \in \{\frac{1}{4}, 1\}$, $P_{sys}^{MS} = 0.13\%$ and $max(U_{LC}^{LO}) = 32.60\%$, while for our proposed scheme, $P_{sys}^{MS} = 6.61\%$ and $max(U_{LC}^{LO}) = 82.45\%$. On the other hand, using a smaller lower-bound value for λ like $\frac{1}{32}$, increases the maximum utilization of the LC tasks with high mode switching probability. For instance, if $U_{HC}^{HI} = 0.8$, then $P_{sys}^{MS} = 92.97\%$ and $max(U_{LC}^{LO}) = 86.43\%$. 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. 5 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 85.29% compared to the existing approaches, while P_{sys}^{MS} is bounded by 9.11% in the worst-case scenario.

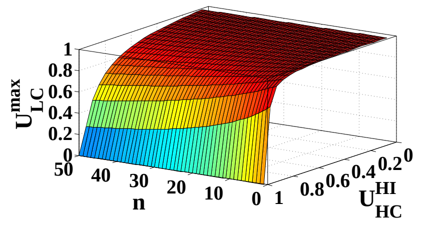
D. Evaluating scheduling approaches under proposed scheme

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.

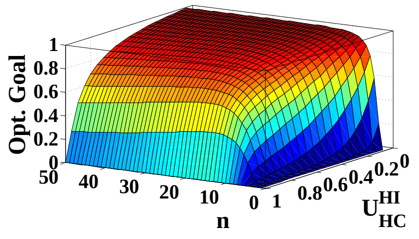
Fig. 6 shows the acceptance ratio for two state-of-the-art scheduling approaches [1], [2], which are improved with our scheme in all utilization bounds. As shown in this figure, when $U_{bound} \leq 0.7$, all tasks sets are schedulable with Liu's



(a) P_{sys}^{MS} by varying n and U_{HC}^{HI}

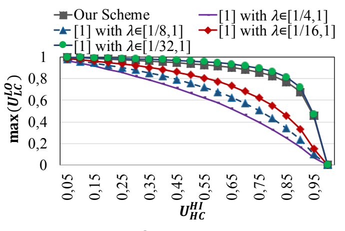


(b) $max(U_{LC}^{LO})$ by varying n and U_{HC}^{HI}

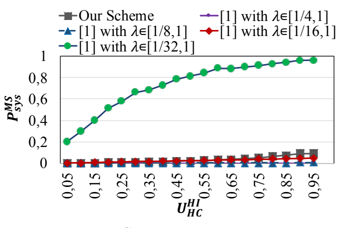


(c) optimization goal

Fig. 3: Effect of n and HC tasks' utilization on maximum assigned utilization to LC tasks and mode switching probability.



(a) $max(U_{LC}^{LO})$ by varying U_{HC}^{HI}



(b) P_{sys}^{MS} by varying U_{HC}^{HI}

Fig. 4: The effectiveness of our proposed scheme in comparison with other policies, proposed in other research works

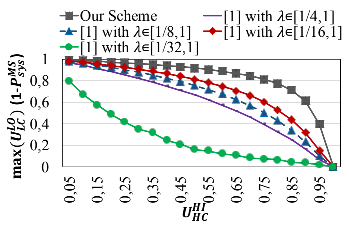


Fig. 5: $max(U_{LC}^{LO})$ and P_{sys}^{MS} by varying U_{HC}^{HI}

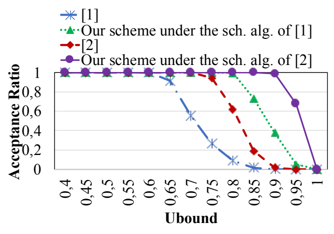


Fig. 6: Different scheduling approaches with our proposed scheme

approach [2], and our scheme. When the system utilization is increased ($0.7 < U_{bound} \leq 0.9$), all task sets are schedulable by our proposed scheme, while the acceptance ratio of Liu's approach [2] is decreased, so that no task set is schedulable for $U_{bound} \geq 0.9$. 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.

VI. Conclusion and Future Work

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 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 the *Chebyshev theorem*. The proposed scheme improves the system utilization and schedulability up to 85.29% and 99.73%, respectively, while bounding the mode switching probability to 9.11% in the worst-case scenario.

As future research, we would extend our scheme for systems with more than two criticality levels. Based on that, we would present a scheduling algorithm and the optimization problem to execute the lower-criticality tasks in higher modes. In addition, we would evaluate the approach with real benchmarks on hardware platforms to show its efficiency.

References

[1] S. Baruah *et al.*, "The Preemptive Uniprocessor Scheduling of Mixed-Criticality Implicit-Deadline Sporadic Task Systems," in *RTNS*, 2012, pp. 145–154.

[2] D. Liu *et al.*, "EDF-VD Scheduling of Mixed-Criticality Systems with Degraded Quality Guarantees," in *RTSS*, 2016, pp. 35–46.

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

[4] Z. Guo *et al.*, "Uniprocessor Mixed-Criticality Scheduling with Graceful Degradation by Completion Rate," in *RTSS*, 2018, pp. 373–383.

[5] B. Ranjbar *et al.*, "FANTOM: Fault Tolerant Task-Drop Aware Scheduling for Mixed-Criticality Systems," *IEEE Access*, vol. 8, 2020.

[6] R. Wilhelm *et al.*, "The Worst-Case Execution-Time Problem—Overview of Methods and Survey of Tools," *TECS*, vol. 7, no. 3, 2008.

[7] A. Kumar *et al.*, "Iterative Probabilistic Performance Prediction for Multi-Application Multiprocessor Systems," *TCAD*, vol. 29, no. 4, 2010.

[8] C. Ballabriga *et al.*, "OTAWA: an open toolbox for adaptive WCET analysis," in *FDL*. Springer, 2010, pp. 35–46.

[9] D. Liu *et al.*, "Scheduling Analysis of Imprecise Mixed-Criticality Real-Time Tasks," *TC*, vol. 67, no. 7, pp. 975–991, 2018.

[10] G. Chen *et al.*, "Utilization-Based Scheduling of Flexible Mixed-Criticality Real-Time Tasks," *TC*, vol. 67, no. 4, pp. 543–558, 2018.

[11] B. Ranjbar *et al.*, "Power-Aware Run-Time Scheduler for Mixed-Criticality Systems on Multi-Core Platform," *TCAD*, 2020.

[12] C. Gu *et al.*, "Partitioned mixed-criticality scheduling on multiprocessor platforms," in *DATE*, 2014, pp. 1–6.

[13] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *RTSS*, 2007, pp. 239–243.

[14] Z. Al-bayati *et al.*, "A four-mode model for efficient fault-tolerant mixed-criticality systems," in *DATE*, 2016, pp. 97–102.

[15] X. Gu and A. Easwaran, "Dynamic Budget Management with Service Guarantees for Mixed-Criticality Systems," in *RTSS*, 2016, pp. 47–56.

[16] 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.

[17] L. Santinelli and L. George, "Probabilities and mixed-criticalities: the probabilistic c-space," in *RTSS*, 2015.

[18] D. Maxim *et al.*, "Probabilistic Analysis for Mixed Criticality Systems Using Fixed Priority Preemptive Scheduling," in *RTNS*, 2017, p. 237–246.

[19] S. Jiménez Gil *et al.*, "Open Challenges for Probabilistic Measurement-Based Worst-Case Execution Time," *ESL*, vol. 9, no. 3, pp. 69–72, 2017.

[20] F. Reghenzani *et al.*, "Dealing with Uncertainty in PWCET Estimations," *TECS*, vol. 19, no. 5, 2020.

[21] A. Bhuiyan *et al.*, "Optimizing Energy in Non-Preemptive Mixed-Criticality Scheduling by Exploiting Probabilistic Information," *TCAD*, vol. 39, no. 11, pp. 3906–3917, 2020.

[22] B. Hu *et al.*, "FFOB: Efficient online mode-switch procrastination in mixed-criticality systems," *Real-Time Systems*, vol. 55, no. 3, pp. 471–513, 2019.

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

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

[25] F. Fortin *et al.*, "DEAP: Evolutionary algorithms made easy," *JMLR*, 2012.

[26] M. Bazzaz *et al.*, "An accurate instruction-level energy estimation model and tool for embedded systems," *TIM*, vol. 62, no. 7, 2013.