Motivating the Use of Machine-Learning For Improving Timing Behaviour of Embedded Mixed-Criticality Systems

Vikash Kumar*†, Behnaz Ranjbar*, and Akash Kumar*, *Senior Member, IEEE*
*Chair of Processor Design, CFAED, Technische Universität Dresden, Dresden, Germany
† Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, India
kvikash@iisc.ac.in, {behnaz.ranjbar, akash.kumar}@tu-dresden.de

***Abstract*—In Mixed-Criticality (MC) systems, due to encountering multiple Worst-Case Execution Times (WCETs) for each task corresponding to the system operation modes, estimating appropriate WCETs for tasks in lower-criticality (LO) modes is essential to improve the system's timing behavior. While numerous studies focus on determining WCET in the high-criticality mode, determining the appropriate WCET in the LO mode poses significant challenges and has been addressed in a few research works due to its inherent complexity. This article introduces a novel scheme to obtain appropriate WCET for LO modes. We propose an ML-based approach for WCET estimation based on the application's source code analysis and the model training using a comprehensive data set. The experimental results show a significant improvement in utilization by up to 23.3% for the ML-based approach, while mode switching probability is bounded by 7.19% in the worst-case scenario.**

***Index Terms*—Machine Learning, Mixed-Criticality, Resource Utilization, Mode Switching Probability, WCET Analysis.**

I. Introduction

Mixed-Criticality (MC) systems are widely employed in embedded systems to effectively address the cost, space, and energy-efficiency demands of diverse applications, including automotive, medical devices, and avionics [1]–[4]. MC systems are purposefully designed to handle multiple tasks with varying criticality levels. The ultimate goal is to prevent catastrophic damages by ensuring that all High-Criticality (HC) tasks are executed correctly before their deadlines. Concurrently, a multitude of Low-Criticality (LC) tasks are efficiently scheduled to maximize processor utilization and achieve the high Quality-of-Service (QoS) [2]–[5].

In conventional real-time systems, tasks are scheduled based on their pessimistic Worst-Case Execution Time (WCET) [6], which can be estimated through various methods, including measurement-based, static analysis, and hybrid approaches. Besides, there are many open-source and commercial tools and frameworks, to determine the pessimistic WCET ($WCET^{pess}$) [6]. However, most tasks' execution times are shorter than conservative WCET, which leads to poor processor utilization and QoS [4],[7],[8]. In this regard, multiple WCETs are defined in MC systems corresponding to the different criticality levels and the ongoing operational mode [9]. Since there are different operational modes in MC systems, initially, an MC system starts its operation in the low-criticality mode (LO mode) while executing the tasks based on the optimistic WCET ($WCET^{opt}$, which is less than $WCET^{pess}$). If the execution time of at least one HC task exceeds its $WCET^{opt}$, the system mode changes from LO to high-criticality mode (HI mode). In such a scenario, all or some LC tasks must be dropped/degraded to provide the processor computation capacity for running the HC tasks and guarantee their correct execution before their deadlines. However, it can drastically affect the service and cause significant performance loss of LC tasks, i.e., QoS degradation. When the gap between the $WCET^{opt}$ and $WCET^{pess}$ is large, more tasks, such as LC tasks, are scheduled at design-time. However, this can cause frequent system mode switches and, consequently, drop more LC tasks at run-time. When this gap is small, the overall processor utilization decreases due to scheduling fewer tasks at design-time [3],[7]. As can be realized, determining an appropriate value of $WCET^{opt}$ for each task is essential in the efficient design of MC systems, which we deal with in this article.

Most previous research works set $WCET^{opt}$ as fraction of $WCET^{pess}$ [1],[9],[10]. These approaches may lead to poor processor utilization (while $WCET^{opt}$ is close to $WCET^{pess}$) or more mode switches (when there is a high gap between $WCET^{opt}$ and $WCET^{pess}$). Besides, researchers in [3],[7] have determined the $WCET^{opt}$ based on the application's average execution cycle (AVG) by using Chebyshev's theorem; however, the mode switching probability is determined pessimistically, which causes poor utilization. Moreover, in [4], Machine-Learning (ML) techniques are employed to determine $WCET^{opt}$ at run-time if there is enough dynamic slack on the processor. However, this approach 1) has run-time timing overhead, and 2) improvement percentage depends on whether a dynamic slack would be generated at run-time.

A. Motivational Example

Fig. 1 shows the execution time distribution of a real application, *ns*, a search application in the multi-dimensional array from Mälardalen benchmark [11], running on the Raspberry Pi 4 board. The X-axis represents the processor clock cycles, and the Y-axis represents the frequency distribution. The $WCET^{pess}$ for this application is 52,531 cycles, using the SWEET tool [12]. With setting the $WCET^{opt}$ as a per-

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 (TU) Dresden.

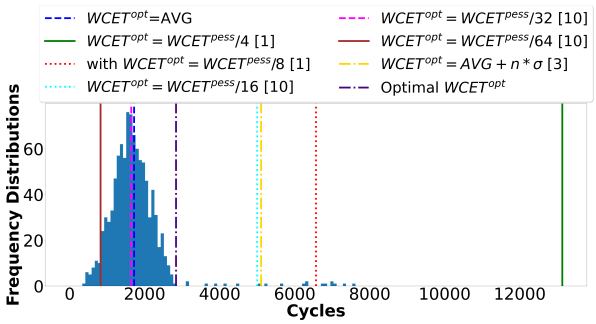

Fig. 1: Execution time distribution and different obtained $WCET^{opt}$ values for an application from Mälardalen benchmark [11], running on the Raspberry Pi 4 board

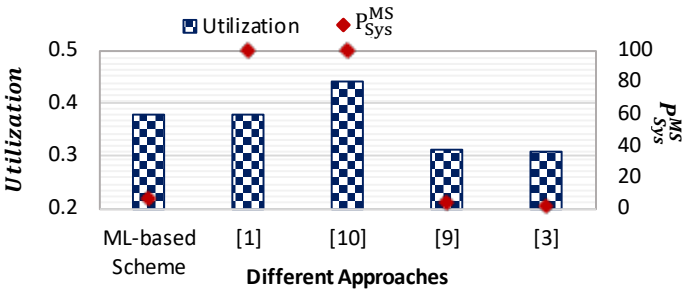

Fig. 2: Analyzing the mode switching probability and maximum utilization that can be assigned to LC tasks for different approaches

centage of $WCET^{pess}$, many system mode switches occur if we set WCET^{opt} to $\frac{WCET^{pess}}{32}$ and $\frac{WCET^{pess}}{64}$, but more LC tasks can be scheduled in the system. On the other hand, if $WCET^{opt}$ is set to $\frac{WCET^{pess}}{4}$ and $\frac{WCET^{pess}}{8}$, or based on the average execution cycle, mode switches are reduced, but poor utilization occurs due to scheduling fewer LC tasks. However, if $WCET^{opt}$ value is close to the indigo line (shown by Optimal $WCET^{opt}$ in the figure), both processor utilization and mode switches can be improved compared to other approaches.

B. Proposed Approach

To tackle this challenge, we propose a novel scheme to obtain the appropriate $WCET^{opt}$ for HC tasks to improve resource utilization while reducing the number of mode switches at design-time. In this scheme, we propose an ML-based approach (which can be generalized to any embedded application) to evaluate the model functionality and performance based on the generated data sets to train and validate different prediction techniques. In the proposed ML-based approach, the training phase is performed once per target architecture. After training, the various ML algorithms can be used to determine $WCET^{opt}$ for different benchmarks. $WCET^{opt}$ is obtained from the best-chosen prediction model from various trained ML models.

To the best of our knowledge, this is the first work that obtains $WCET^{opt}$, utilizing ML models while guaranteeing real-timeliness, improving the utilization by scheduling more LC tasks and reducing the mode switches with no timing overhead at run-time.

C. Evaluation

We evaluate the proposed scheme with extensive experiments on a real board, Raspberry Pi, by executing 1000 instances of seven applications from [11]. Fig. 2 shows the mode switching probability of the system and the maximum utilization that can be assigned to LC tasks if we set WCET^{opt} according to different approaches. These two objectives, i.e., mode switching probability and LC task utilization, are computed based on what is proposed in [3]. We observe that in [9] and [3], the mode switching probability is less, while the utilization decreases due to choosing the WCET^{opt} too pessimistic. Besides, in [1] and [10], the processor utilization increases, while the mode switching probability is higher due to choosing WCET^{opt} in an optimistic manner. Our approach performs better than others in achieving a reasonable trade-off between mode switching and utilization. As can be seen, our approaches have higher utilization and fewer mode switches due to obtaining an appropriate WCET^{opt} for each HC task and scheduling more LC tasks in the systems. As a result, our ML-based approach can improve the utilization by up to 23.3%, while mode switching probability is bounded by 7.19% in the worst-case scenario.

II. Conclusion and Future Work

This paper proposed a novel scheme that utilizes ML models to obtain optimistic WCET for mixed-criticality tasks. The model estimates optimistic WCET from the applications' source code. The proposed scheme improves processor utilization significantly while reducing the system mode switches.

In future work, we would extend our scheme by analyzing different ML models. We would also analyze the existing WCET determination approaches to choose one of them for better training and objective optimization.

References

[1] S. Baruah *et al.*, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 2012.

[2] A. Burns *et al.*, "A survey of research into mixed criticality systems," *ACM Computing Surveys (ACM CSUR)*, vol. 50, no. 6, 2017.

[3] B. Ranjbar *et al.*, "Improving the timing behaviour of mixed-criticality systems using chebyshev's theorem," in *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[4] ——, "ADAPTIVE: Agent-Based Learning for Bounding Time in Mixed-Criticality Systems," in *Proc. of Design Automation Conference (DAC)*, 2023.

[5] ——, "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," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 7, no. 3, 2008.

[7] B. Ranjbar *et al.*, "BOT-MICS: Bounding Time Using Analytics in Mixed-Criticality Systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 10, 2022.

[8] ——, "Learning-oriented qos-and drop-aware task scheduling for mixed-criticality systems," *Computers*, vol. 11, no. 7, 2022.

[9] D. Liu *et al.*, "Scheduling analysis of imprecise mixed-criticality real-time tasks," *IEEE Transactions on Computers (TC)*, vol. 67, no. 7, 2018.

[10] Z. Guo *et al.*, "Uniprocessor mixed-criticality scheduling with graceful degradation by completion rate," in *Proc. of Real-Time Systems Symposium (RTSS)*, 2018.

[11] J. Gustafsson *et al.*, "The mälardalen wcet benchmarks: Past, present and future," in *Proc. of International Workshop on Worst-Case Execution Time Analysis (WCET)*, 2010.

[12] B. Lisper, "Sweet–a tool for wcet flow analysis," in *Proc. of International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, 2014.