Journal of Systems Architecture 160 (2025) 103361

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

EDF-based Energy-Efficient Probabilistic Imprecise Mixed-Criticality Scheduling

Yi-Wen Zhang *, Jin-Long Zhang

College of Computer Science and Technology, Huaqiao University, Xiamen, 361021, China

ARTICLE INFO

Keywords:
Imprecise Mixed-Criticality
Energy management
DVFS
Probabilistic schedulability

ABSTRACT

We focus on Mixed-Criticality Systems (MCS), which involves the integration of multiple subsystems with varying levels of criticality on shared hardware platforms. The classic MCS task model assumes hard real-time constraints and no Quality-of-Service (QoS) for low-criticality tasks in high-criticality mode. Many researchers have put forward a range of extensions to the classic MCS task model to make MCS theory more applicable in industry practice. In this paper, we consider an Imprecise MCS taskset scheduled with Earliest Deadline First algorithm on a uniprocessor platform, and propose an Energy-Efficient Task Execution Model that guarantees (deterministic or probabilistic) schedulability, allows degraded QoS to low-criticality tasks in high-criticality mode, and applies Dynamic Voltage and Frequency Scaling to save energy.

1. Introduction

Mixed-Criticality Systems (MCS) [1] involve the integration of multiple sub-systems with varying criticality levels on a shared hardware platform. For example, the automotive safety certification standard ISO 26262 and the avionics safety certification standard DO-178C. Since the introduction of the MCS concept by Vestal [2], there has been considerable research conducted on this topic [1,3,4]. Many researchers have put forward a range of extensions to the classic MCS task model to make MCS theory more applicable in industry practice, including:

- To reduce the pessimism in task worst-case execution time (WCET) estimation and system schedulability analysis, researchers have proposed probabilistic schedulability analysis techniques where the task WCETs (and/or periods) are represented by random variables, and the system is allowed to miss deadlines with a small probability [5].
- The original assumption that all low-criticality (LO) tasks are discarded in high-criticality (HI) mode is likely to be undesirable in industry practice, hence researchers have proposed various approaches to allow a certain level of degraded Quality-of-Service (QoS) to LO tasks in HI mode [1].
- To address energy-constrained safety–critical systems, researchers have proposed power and energy-aware scheduling algorithms with *Dynamic Voltage and Frequency Scaling (DVFS)* for MCS [6].

In this paper, we consider all the above different aspects within a unified framework. We consider an Imprecise MCS probabilistic taskset scheduled with Earliest Deadline First (EDF) algorithm on a uniprocessor platform, and propose an Energy-Efficient Task Execution Model that guarantees (deterministic or probabilistic) schedulability, allows degraded QoS to LO tasks in HI mode, and applies DVFS to save energy. Although the work in [7] is the closest to ours, there are several key differences. Firstly, it schedules tasks under non-preemptive fixed-priority (NPFP) [8] scheduling policy while our work schedules tasks with a preemptive EDF. Secondly, it uses probabilistic WCET (pWCET) to determine the probability of mode transition and uses a deterministic schedulability analysis while our work includes deterministic or probabilistic schedulability analysis. Finally, it uses the response time analysis to determine the schedulability analysis while our work uses Demand Bound Function (DBF) to determine the schedulability analysis. In short, the work is first to address the energy issue and schedulability test of the Imprecise MCS probabilistic taskset MCS taskset scheduling under EDF.

The remainder of the paper is organized as follows. We present background and related work in Section 2. Section 3 presents preliminaries. Section 4 presents our probabilistic IMC scheduling; Section 5 presents the Energy-Efficient Task Execution Model; Section 6 presents experimental results; Section 7 discusses practical issues. Finally, Section 8 presents conclusions and future work.

* Corresponding author.
E-mail addresses: zyw@hqu.edu.cn (Y.-W. Zhang), sang_yunl@stu.hqu.edu.cn (J.-L. Zhang).

https://doi.org/10.1016/j.sysarc.2025.103361
Received 11 September 2024; Received in revised form 3 February 2025; Accepted 4 February 2025
Available online 12 February 2025
1383-7621/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

2. Background and related work

2.1. Background and motivation

Resource-constrained embedded systems. In order to motivate the need for probabilistic scheduling and DVFS addressed in this paper, we first discuss the issue of hardware resource constraints in real-time embedded systems, including but not limited to MCS, which are especially pertinent for mass-produced consumer products such as ground vehicles and drones (Unmanned Aerial Vehicles), due to monetary cost as well as Size, Weight, and Power (SWaP) constraints. Automotive Electrical/Electronic (E/E) systems typically have stringent hardware resource constraints. In modern high-end vehicles, there can be up to 100 ECUs (Electronic Control Units) embedded within them, and each model can be sold millions of times. An overall savings of millions of dollars may be achieved by saving a few dollars per ECU. Hence, a designer of E/E systems should choose the cheapest ECU according to their application's needs. The monetary cost pressure on relatively cheap consumer drones is even higher. Next, let us consider the issue of SWaP, which lumps together three factors that are closely correlated due to the same underlying cause of hardware resource constraints. The significance of SWaP is obvious in battery-powered mobile devices like drones and mobile robots, where operating time and physical constraints are limited. However, SWaP considerations are equally applicable to ground vehicles that are equipped with sizable battery systems. Electronics within autonomous vehicles consume substantial power, impacting the range of electric vehicles or the fuel consumption of gasoline vehicles. Size and weight affect consumer acceptance, e.g., an autonomous vehicle with a trunk full of electronics is not likely to be acceptable to the average consumer. The issue of significant hardware resource constraints in MCS has motivated a line of work on processing and memory resource optimization algorithms for MCS [9].

Motivation for probabilistic schedulability analysis. Recently, Akesson et al. [10] investigated 120 industry practitioners in real-time embedded systems, and results indicated that soft or firm real-time constraints are prevalent even in safety–critical application domains. A minority (15%) of the surveyed systems were considered strictly hard real-time (no deadlines to be missed). Thus, designing the timing behavior of a system function to ensure a much lower failure rate did not affect the system's total schedulability.

Industry safety certification standards specify acceptable failure rates depending on the system's criticality levels such as each ASIL has a permitted failure probability of 10^{-9} for ASIL D, 10^{-8} for ASIL C and B, and 10^{-7} for ASIL A in the automotive standard ISO-26262 [5]. Relaxing the hard real-time assumption can help reduce pessimism in task WCET estimation and system schedulability analysis and increase schedulable utilization significantly. Von der Brüggen et al. [11] demonstrated large gains in processor utilization with experiments using randomly-generated workloads, e.g., a gain of at least 12% schedulable utilization for an acceptable worst-case deadline failure probability of 10^{-6}. This motivates probabilistic schedulability analysis as an effective technique for reducing analysis pessimism and increase processor utilization in resource-constrained embedded systems.

Motivation for not dropping LO tasks in HI mode. Consider the automotive standard ISO-26262, where ASIL determination of hazardous events is based on three parameters: "severity", "probability of exposure" and "controllability". An individual's vulnerability to harm in a potentially hazardous situation is determined by severity. Probability is the likelihood that harm will occur, while controllability is the ability to avoid harm or damage through prompt action by the agents involved (e.g. a driver of the vehicle). It cannot always be assumed that a software function that is part of a high ASIL functionality is more important than one that is part of a lower ASIL functionality, as both may be safety–critical, and each function's failure may cause severe damage [12].

2.2. The classic MCS task model

The MCS taskset Γ includes n independent sporadic tasks $\Gamma = \{\tau_i | 1 \leq i \leq n\}$ [13,14]. Although there may be multiple (4–5) criticality levels in general, we present the task model assuming a dual-criticality system with criticality levels LO and HI for the sake of simplicity. The taskset Γ includes two subsets: LO tasks $\Gamma_{LO} = \{\tau_i \in \Gamma | L_i = LO\}$ and HI tasks $\Gamma_{HI} = \{\tau_i \in \Gamma | L_i = HI\}$. Each task $\tau_i \in \Gamma$ is described by $(L_i, T_i, D_i, C_i^{LO}, C_i^{HI})$:

- $L_i \in \{LO, HI\}$ denoted its criticality level.
- T_i denoted its period.
- D_i denoted its relative deadline.
- C_i^{LO} denoted its WCET in LO mode.
- C_i^{HI} denoted its WCET in HI mode for HI tasks ($L_i = HI$), with $C_i^{HI} \geq C_i^{LO}$.

Task execution model of classic MCS. The system is first initialized to be in LO mode. LO tasks $\tau_i \in \Gamma_{LO}$ are monitored at run time and their execution is no more than their C_i^{LO}. The system is schedulable in LO mode if all tasks $\tau_i \in \Gamma$ can complete their LO mode WCETs C_i^{LO} within their respective deadlines. If any HI task $\tau_i \in \Gamma_{HI}$ executes beyond its C_i^{LO}, the system enters HI mode while all LO tasks in Γ_{LO} are abandoned. The system is schedulable in HI mode if all HI tasks $\tau_i \in \Gamma_{HI}$ can complete their HI mode WCETs C_i^{HI} within their respective deadlines. The system switches back to LO mode at an idle instant if no jobs wait for executions at this time [15]. The system is schedulable if both modes are schedulable.

The state-of-the-art scheduling algorithms for the classic MCS task model include Fixed-Priority scheduling [14], and Earliest-Deadline First with Virtual Deadline (EDF-VD) [16] for Dynamic-Priority scheduling on uniprocessor systems. Subsequently, many extensions to the classic MCS task model have been proposed, as discussed next.

2.3. Degraded QoS for LO tasks

The degraded QoS of LO tasks in HI mode is achieved by decreasing execution time budgets [17] or adding the task period [18] for LO tasks.

Liu et al. [17] proposed the Imprecise Mixed-Criticality (IMC) task model in which a HI task τ_i ($L_i = HI$) is assigned a greater estimated WCET compared to its estimation in LO mode ($C_i^{LO} \leq C_i^{HI}$), while a LO task τ_i ($L_i = LO$) is assigned a smaller estimated WCET in HI mode compared to the estimation in LO mode ($C_i^{LO} \geq C_i^{HI}$). They considered EDF-VD scheduling on a single processor system, and presented two schedulability tests, one based on the utilization bound test, and the other based on the Demand Bound Function (DBF). Davis et al. [19] addressed the IMC task model under fixed-priority scheduling, and presented a Compensating AMC Scheduling scheme and two schedulability tests. Jiang et al. [20] presented a concrete implementation of the IMC task model in the form of a configurable processor floating point unit hardware design, as well as schedulability analysis and optimized priority assignment algorithms based on fixed-priority scheduling.

2.4. Energy-aware scheduling for MCS

DVFS dynamically adjusts the processor supply voltage and speed (frequency) based on the system's workload, which is an effective energy-saving technique [21]. Most modern microprocessors, including those used in embedded systems, provide support for DVFS. Our recent survey paper [6] provided an overview of recent developments in energy-aware real-time scheduling for MCS, predominantly focusing on DVFS.

Recently, power and energy-aware real-time scheduling for MCS has attracted significant attention [6]. Huang et al. [22] proposed a scheduling algorithm for MCS based on EDF-VD [16]. This scheduling algorithm reduces energy consumption by optimizing virtual deadlines

and processor speeds. Zhang [23] used the dynamic slack time generated from late arrival tasks to reduce energy consumption. This work is extended to MCS with fixed-priority preemptive scheduling [24] and dynamic priority non-preemptive scheduling [25]. Zhang et al. [26] tackled the issue of MCS with shared resources and proposed a dual-speed scheduling algorithm. This algorithm ensured both the system schedulability and mutually exclusive access to shared resources. However, it assumed that all tasks execute with their WCET. Zhang [27] used the difference between the actual execution time and WCET to save energy. These works focus on the classic MCS task model. Zhang [28] focused on the IMC task model in which LO tasks allow Qos in HI mode and proposed an energy-aware scheduling algorithm (EA-IMC).

There has been a small number of recent works on energy-aware MCS on multiprocessors. Narayana et al. [29] considered the energy minimization problem for multiprocessor MCS based on DVFS. They first proposed an optimal solution and an effective lightweight heuristic on a uniprocessor, then extended these results to multicore systems. Ranjbar et al. [30] proposed a heuristic algorithm for online peak power and thermal management of a multicore MCS by using the slack time and per-cluster DVFS. Recently, some researchers [31] studied the IMC task model on multiprocessors in which LO tasks allow QoS in HI mode and proposed the partitioned scheduling algorithm. In addition, this work is extended to shared resource scheduling [32]. However, the above studies assume that tasks execute with their deterministic WCET.

2.5. Probabilistic scheduling for MCS

Santinelli and George [33] presented an initial solution to probabilistic schedulability analysis for EDF scheduling of MCS based on the concept of probabilistic C-Space. Maxim et al. [34] presented a probabilistic fixed-priority schedulability analysis [14]. Singh et al. [35] considered a novel MCS task model with job-level mode switching, and presented a graph-traversal-based analytic framework for non-preemptive job-level fixed-priority probabilistic schedulability analysis. Draskovic et al. [36] proposed metrics that are inspired by industry safety standards, including the probability of deadline miss per hour, the expected time before degradation happens, and the duration of the degradation, and presented a system-wide approach to probabilistic scheduling of MCS. Guo et al. [37] proposed a new task model in which a new parameter is added to characterize the distribution of the WCET estimations for each task. They presented efficient algorithms for MCS scheduling under this task model for both independent tasks and failure-dependent tasks.

We are aware of only one related work that addressed energy-aware scheduling in MCS assuming probabilistic task execution times. Bhuiyan et al. [7] proposed a probabilistic technique to derive an energy-efficient processor speed that minimized the average energy consumption with DVFS, while ensuring deadlines of all tasks in MCS. This work used non-preemptive fixed-priority scheduling and deterministic schedulability test based on Worst-Case Response Time analysis, instead of probabilistic schedulability analysis. It is not directly comparable to our work due to the different task models and analysis techniques.

Table 1 summarized related work on probabilistic Scheduling for MCS.

Table 1
Related work on probabilistic Scheduling for MCS. Abbreviations: Prob. (Probabilistic); S.A. (Schedulability Analysis).

Work	Sched. Algo.	Prob. S.A.	Energy-Aware	LO tasks dropped in HI Mode
Santinelli and George (2015) [33]	EDF	Y	N	Y
Maxim et al. (2017) [34]	FP	Y	N	Y
Singh et al. (2020) [35]	NPFP	Y	N	Y
Draskovic et al. (2021) [36]	FP	Y	N	N
Guo et al. (2021) [37]	EDF	Y	N	Y
Bhuiyan et al. (2020) [7]	NPFP	N	Y	Y
This work	EDF	Y	Y	N

3. Preliminaries

3.1. Task model

Our task model is inspired by the IMC task model [17], with extensions to the probabilistic scheduling scenario. We first introduce some basic notations for probabilistic scheduling. A task τ_i's *probabilistic WCET (pWCET)* C_i is a random variable characterized by a Probability Mass Function (PMF) $f_{C_i}(\cdot)$, where $f_{C_i}(et) = P(C_i = et)$ denotes the probability that its WCET is equal to et.[1] Given the PMF $f_{C_i}(\cdot)$, we can easily obtain the corresponding Cumulative Distribution Function (CDF) $F_{C_i}(\cdot)$, where $F_{C_i}(et) = P(C_i \le et) = \sum_{x \le et} f_{C_i}(x)$. The Complementary Cumulative Distribution Function (1-CDF) is defined as $\bar{F}_{C_i}(et) = P(C_i > et) = 1 - F_{C_i}(et)$.

We consider the MCS taskset Γ including n independent periodic tasks $\Gamma = \{\tau_i | 1 \le i \le n\}$ scheduled with preemptive EDF on a single processor platform. (It is a special case of EDF-VD with a deadline scaling factor $x = 1$.) We assume a dual-criticality system with criticality levels LO and HI for the sake of simplicity. The taskset Γ consists of two subsets: LO tasks $\Gamma_{LO} = \{\tau_i \in \Gamma | L_i = LO\}$ and HI tasks $\Gamma_{HI} = \{\tau_i \in \Gamma | L_i = HI\}$. Each task $\tau_i \in \Gamma$ is described by a tuple of parameters $\langle L_i, T_i, D_i, C_i, C_i^{LO}, C_i^{HI}, C_i^{deg} \vee C_i^{thr} \rangle$:

- $L_i \in \{LO, HI\}$ denotes its criticality level.
- T_i denotes its period.
- D_i denotes its constrained deadline ($D_i \le T_i$).
- C_i is its nominal pWCET, a discrete random variable with K discrete values characterized by PMF $f_{C_i}(\cdot)$ and CDF $F_{C_i}(\cdot)$. It has the minimum value C_i^{min} with index $ind(C_i^{min}) = 0$ and maximum value C_i^{max} with index $ind(C_i^{max}) = K - 1$ among the K discrete values of C_i.
- C_i^{LO} is its pWCET in LO mode, characterized by PMF $f_{C_i^{LO}}(\cdot)$ and CDF $F_{C_i^{LO}}(\cdot)$.
- C_i^{HI} is its pWCET in HI mode, characterized by PMF $f_{C_i^{HI}}(\cdot)$ and CDF $F_{C_i^{HI}}(\cdot)$.
- C_i^{deg} is valid for LO tasks ($L_i = LO$), and denotes its Degraded WCET in HI mode C_i^{deg} with index $ind(C_i^{deg}) \in [0, K-1]$.
- C_i^{thr} is valid for HI tasks ($L_i = HI$), and denotes its Threshold WCET in LO mode C_i^{thr} with index $ind(C_i^{thr}) \in [0, K-1]$.

Task execution model. The system is first initialized to be in LO mode. If any HI task $\tau_i \in \Gamma_{HI}$ executes beyond its C_i^{thr}, the system switches from LO mode to HI mode. At the mode switch instant t_s, if jobs of LO tasks have run for longer than their C_i^{deg}, any such jobs will be dropped, without suppressing future arrivals thereof. In addition, if a LO job has executed for less than C_i^{deg} by the switch time instant, these carry-over jobs that have an arrival time before t_s and have absolute deadlines after t_s will continue to execute the leftover execution up to C_i^{deg}. While in HI mode, each LO task $\tau_i \in \Gamma_{LO}$ executes no more than its C_i^{deg}, i.e., it is dropped if its execution time exceeds C_i^{deg}. The system switches from HI mode to LO mode at an idle instant if no jobs wait for executions at this time. Moreover, incomplete tasks are dropped at their deadlines, hence there does not exist a backlog of outstanding execution at the end of each hyper-period (this is a common assumption in industry practice [10].

The pWCET of a LO task in LO mode, or the pWCET of a HI task in HI mode, is the same as its nominal pWCET C_i. The pWCET of a HI

[1] Calligraphic letters are used to represent distributions while non calligraphic letters are for scalars.

task τ_i in LO mode is trimmed with the upper bound C_i^{thr} to have the conditional PMF $f_{C_i^{LO}}(et) = P(C_i = et \mid et \le C_i^{thr})$. The pWCET of a LO task τ_i in HI mode is trimmed with the upper bound C_i^{deg} to have the conditional PMF $f_{C_i^{HI}}(et) = P(C_i = et \mid et \le C_i^{deg})$. In other words, C_i^{deg} is LO task τ_i's execution time budget in HI mode, and C_i^{thr} is HI task τ_i's execution time budget in LO mode. This is inspired by the IMC task model [17,19,20]. They are computed with Eqs. (1) and (2):

$$\forall \tau_i \in \Gamma_{LO} : f_{C_i^{LO}}(et) = f_{C_i}(et), \tag{1}$$

$$f_{C_i^{HI}}(et) = \begin{cases} \sum_{et' \ge C_i^{deg}} f_{C_i^{LO}}(et'), & et = C_i^{deg} \\ f_{C_i^{LO}}(et), & et < C_i^{deg} \\ 0, & et > C_i^{deg} \end{cases}$$

$$\forall \tau_i \in \Gamma_{HI} : f_{C_i^{HI}}(et) = f_{C_i}(et) \tag{2}$$

$$f_{C_i^{LO}}(et) = \begin{cases} \sum_{et' \ge C_i^{thr}} f_{C_i^{HI}}(et'), & et = C_i^{thr} \\ f_{C_i^{HI}}(et), & et < C_i^{thr} \\ 0, & et > C_i^{thr} \end{cases}$$

Since task τ_i's period T_i is a constant in both LO and HI modes, its *probabilistic Worst-Case Utilization (pWCU)* can be obtained by dividing its pWCET by its period: $\mathcal{U}_i = C_i/T_i$, $\mathcal{U}_i^{LO} = C_i^{LO}/T_i$ in LO mode, and $\mathcal{U}_i^{HI} = C_i^{HI}/T_i$ in HI mode. The pWCU of a taskset can be obtained by summing the pWCUs of all tasks in the taskset.

Example 1. A taskset Γ_1 with two tasks is shown in Table 2. Each task τ_i's nominal pWCET C_i is shown in matrix form defined in Eq. (3). For the matrix form, the first row denotes each discrete value of C_i; the second row denotes probability values of the PMF $f_{C_i}(\cdot)$; and the third row denotes cumulative probability values of the CDF $F_{C_i}(\cdot)$.

$$\begin{pmatrix} C_0 & C_1 & \dots & C_{K-1} \\ f_{C_i}(C_0) & f_{C_i}(C_1) & \dots & f_{C_i}(C_{K-1}) \\ F_{C_i}(C_0) & F_{C_i}(C_1) & \dots & F_{C_i}(C_{K-1}) \end{pmatrix} \tag{3}$$

The PMF of τ_i's pWCET in LO mode C_i^{LO} is obtained by Eq. (2); the PMF of its pWCET in HI mode C_i^{HI} is obtained by Eq. (1). For the toy example, the LO task τ_1's nominal pWCET C_1 has two possible values 1 and 2, each with probability 0.5; its pWCET in LO mode C_1^{LO} is the same as C_1; its pWCET in HI mode C_1^{HI} is obtained by trimming C_1 with the upper bound $C_1^{deg} = 1$ and $ind(C_1^{deg}) = 0$ (assuming the index starts from 0), with one possible value of 1 with a probability 1.0. The HI task τ_2's nominal pWCET C_2 has two possible values 1 and 2, each with probability 0.5; its pWCET in LO mode C_2^{LO} is obtained by trimming C_2 with the upper bound $C_2^{thr} = 1$ and $ind(C_2^{thr}) = 0$, with one possible value of 1 with a probability 1.0; its pWCET in HI mode C_2^{HI} is the same as C_2. The matrix that denotes τ_i's pWCU is obtained by dividing each term in the first row of its pWCET matrix by its period T_i.

Eq. (4) shows the definitions of pWCU for the subset of LO tasks Γ_{LO} in LO mode. (As mathematical background, the addition of two discrete random variables $\mathcal{X}$ and $\mathcal{Y}$ results in a new random variable $\mathcal{Z}$ with PMF computed by the convolution of the two PMFs $\mathcal{X}$ and $\mathcal{Y}$, i.e., $\mathcal{Z} = \mathcal{X} \bigotimes \mathcal{Y}$, where $P(\mathcal{Z} = z) = \sum_{k=-\infty}^{\infty} P(\mathcal{X} = k)P(\mathcal{Y} = z - k)$.

$$\mathcal{U}_{LO}^{LO}(\Gamma) = \bigotimes_{\tau_i \in \Gamma_{LO}} \mathcal{U}_i^{LO}, \mathcal{U}_{HI}^{HI}(\Gamma) = \bigotimes_{\tau_i \in \Gamma_{HI}} \mathcal{U}_i^{HI}, \tag{4}$$

where $\mathcal{U}_{LO}^{LO}(\Gamma)$ denotes pWCU of Γ_{LO} in LO mode; $\mathcal{U}_{HI}^{HI}(\Gamma)$ denotes pWCU of Γ_{HI} in HI mode.

3.2. Existing deterministic IMC scheduling

Liu et al. [17] have studied the schedulability test for deterministic IMC task model and proposed the sufficient conditions of the schedulability under EDF-VD. We first introduce the following notations.

Table 2
Taskset parameters of Γ_1, with $C_1^{deg} = 1, C_2^{thr} = 1$.

Task	L_i	$T_i = D_i$	C_i	C_i^{LO}	C_i^{HI}	$\mathcal{U}_i^{LO}$	$\mathcal{U}_i^{HI}$
τ_1	LO	2	$\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \\ 0.5 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \\ 0.5 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1.0 \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} 0.5 & 1.0 \\ 0.5 & 0.5 \\ 0.5 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 0.5 \\ 1.0 \\ 1.0 \end{pmatrix}$
τ_2	HI	2	$\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \\ 0.5 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1.0 \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \\ 0.5 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 0.5 \\ 1.0 \\ 1.0 \end{pmatrix}$	$\begin{pmatrix} 0.5 & 1.0 \\ 0.5 & 0.5 \\ 0.5 & 1.0 \end{pmatrix}$

- $[\![A]\!]_0$ stands for $\max(A, 0)$.
- t_s stands for the mode-switch time.
- $m_i = \lfloor \frac{t - D_i}{T_i} \rfloor$ and $k_i = \lfloor \frac{t_s}{T_i} \rfloor$ are the number of jobs for τ_i in the interval $[0, t)$ and $[0, t_s)$, respectively.
- $DBF_L(\tau_i, t)$ stands for the processor demand of any task $\tau_i \in \Gamma$ within $[0, t)$ in LO mode.
- $DBF(J_L, t)$ and $DBF(J_H, t)$ stand for the processor demand of a carry-over job released by task $\tau_i \in \Gamma_{LO}$ and $\tau_i \in \Gamma_{HI}$ within $[0, t)$, respectively.
- r_i stands for the arrival time of the carry-over job that arrives before t_s and has a deadline after t_s.
- $DBF_L^H(\tau_i, t)$ stands for the processor demand of a LO task τ_i within $[0, t)$ in HI mode, while $DBF_H^H(\tau_i, t)$ stands for the processor demand of a HI task τ_i within $[0, t)$ in HI mode.

Fig. 1 illustrates a carry-over job and the mode switch. The downward arrow represents the job arrival time. If the execution time of τ_i exceeds C_i^{LO} without signaling completion, the system switches from LO mode to HI mode. J_H is a carry-over job.

According to the **Task Execution model**, the processor demand of LO carry-over jobs is always less than or equal to C_i^{LO}, while the processor demand of HI carry-over jobs is always less than or equal to C_i^{HI}. Therefore, $DBF(J_L, t)$ can be calculated as follows:

$$DBF(J_L, t) = \begin{cases} C_i^{LO}, & r_i + D_i \le t \\ 0, & otherwise. \end{cases} \tag{5}$$

and $DBF(J_H, t)$ can be calculated as follows:

$$DBF(J_H, t) = \begin{cases} C_i^{HI}, & r_i + D_i \le t \\ 0, & otherwise. \end{cases} \tag{6}$$

From [3,17], we have the following Theorems.

Theorem 1. *A deterministic IMC taskset Γ is schedulable under EDF in LO mode, if $0 < \forall t \le t_{max}$,*

$$\sum_{\tau_i \in \Gamma} DBF_L(\tau_i, t) \le t, \tag{7}$$

where $DBF_L(\tau_i, t) = [\![m_i + 1]\!]_0 \cdot C_i^{LO}$, and t_{max} is a hyper-period.

Theorem 2. *A deterministic IMC taskset Γ is schedulable under EDF in HI mode, if $0 < \forall t \le t_{max}$, $0 < t_s < t$,*

$$\sum_{\tau_i \in \Gamma_{LO}} DBF_L^H(\tau_i, t_s, t) + \sum_{\tau_j \in \Gamma_{HI}} DBF_H^H(\tau_j, t_s, t) \le t, \tag{8}$$

where $DBF_L^H(\tau_i, t_s, t) = k_i C_i^{LO} + DBF(J_L, t) + c_i C_i^{HI}$, and $DBF_H^H(\tau_i, t_s, t)$ can be determined as follows:

$$DBF_H^H(\tau_i, t_s, t) = \begin{cases} DBF(1), & D_i \le t - t_s; \\ \max\{DBF(1), DBF(2)\}, & Otherwise, \end{cases} \tag{9}$$

where $DBF(1) = b_i C_i^{LO} + DBF(J_H, t) + a_i C_i^{HI}$, $DBF(2) = k_i C_i^{LO} + DBF(J_H, t)$, $a_i = [\![m_i - b_i]\!]_0$, $b_i = [\![\lfloor \frac{t_s - (t - D_i - m_i T_i)}{T_i} \rfloor]\!]_0$, and $c_i = [\![m_i - k_i]\!]_0$.

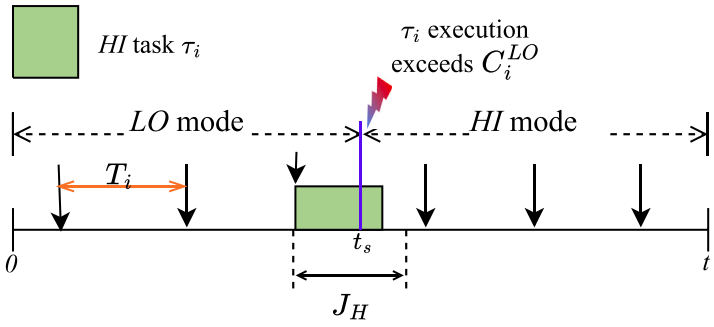


Fig. 1. Carry-over job.

4. Probabilistic IMC scheduling

4.1. Schedulability analysis

Before presenting the schedulability analysis, let us introduce a few notations.

- $\max\{\mathcal{R}\}$ stands for the maximum value of random variable $\mathcal{R}$.
- $\mathcal{V}(x) = \begin{pmatrix} x \\ 1 \\ 1 \end{pmatrix}$, where x is a constant.
- $\mathcal{DBF}_L(\tau_i, t)$ stands for the probabilistic processor demand of any task τ_i within $[0, t)$ in LO mode.
- $\mathcal{DBF}(J_L, t)$ and $\mathcal{DBF}(J_H, t)$ stand for the probabilistic processor demand of a carry-over job released by the task $\tau_i \in \Gamma_{LO}$ and $\tau_i \in \Gamma_{LO}$ within $[0, t)$, respectively.
- $\mathcal{DBF}_L^H(\tau_i, t)$ stands for the probabilistic processor demand of a LO task τ_i within $[0, t)$ in HI mode, while $\mathcal{DBF}_H^H(\tau_i, t)$ stands for the probabilistic processor demand of a HI task τ_i within $[0, t)$ in HI mode.
- $\mathcal{DBF}_L(t)$ stands for the probabilistic processor demand of all tasks within $[0, t)$ in LO mode.
- $\mathcal{DBF}_H(t)$ stands for the probabilistic processor demand of all tasks within $[0, t)$ in HI mode.
- $\Pi_{t=1}^{t_{max}} t = 1 \times 2 \times \cdots \times t_{max}$.

According to [3,17,33], the probabilistic processor demand of any task $\tau_i \in \Gamma$ within $[0, t)$ in LO mode can be calculated as follows:

$$\mathcal{DBF}_L(\tau_i, t) = \mathcal{V}([\![m_i + 1]\!]_0) \odot C_i^{LO}, \tag{10}$$

where $\odot$ denotes the Hadamard product, where each element in the ith row of the right matrix is multiplied by the element on the ith row of the left vector.

In addition, the probabilistic processor demand of all tasks within $[0, t)$ in LO mode can be calculated as follows:

$$\mathcal{DBF}_L(t) = \bigotimes_{\tau_i \in \Gamma} \mathcal{DBF}_L(\tau_i, t). \tag{11}$$

The probabilistic processor demand of a carry-over job released by LO task τ_i within $[0, t)$ can be calculated as follows:

$$\mathcal{DBF}(J_L, t) = \begin{cases} C_i^{LO}, & r_i + D_i \le t \\ \mathcal{V}(0), & otherwise. \end{cases} \tag{12}$$

The probabilistic processor demand of a carry-over job released by HI task τ_i within $[0, t)$ can be calculated as follows:

$$\mathcal{DBF}(J_H, t) = \begin{cases} C_i^{HI}, & r_i + D_i \le t \\ \mathcal{V}(0), & otherwise. \end{cases} \tag{13}$$

The probabilistic processor demand of any task $\tau_i \in \Gamma_{LO}$ within $[0, t)$ in HI mode can be calculated as follows:

$$\mathcal{DBF}_L^H(\tau_i, t) = (\mathcal{V}(k_i) \odot C_i^{LO}) \bigotimes \mathcal{DBF}(J_L, t) \bigotimes (\mathcal{V}(c_i) \odot C_i^{HI}). \tag{14}$$

According to [3,17], we should consider two cases to determine the probabilistic processor demand of any task $\tau_i \in \Gamma_{HI}$ within $[0, t)$ in HI mode.

Case 1: $D_i \le t - t_s$. The maximum demand of a job released by the HI task τ_i is generated while its deadline coincides with t. According to Eq. (9) in Theorem 2, the probabilistic processor demand of any task $\tau_i \in \Gamma_{HI}$ within $[0, t)$ in HI mode is equal to $\mathcal{DBF}(1) = (\mathcal{V}(b_i) \odot C_i^{LO}) \bigotimes \mathcal{DBF}(J_H, t) \bigotimes (\mathcal{V}(a_i) \odot C_i^{HI})$.

Case 2: $D_i > t - t_s$. The HI task τ_i has at most one job with a processor demand C_i^{HI}. If the deadline of this job is D_i, the probabilistic processor demand is the same as $\mathcal{DBF}(1)$. Moreover, the only way to increase the demand of the HI task τ_i is to add a new job in the interval. In other words, the first job of the HI task τ_i arrives at time 0. Therefore, the processor demand includes two parts: one part is the demand of all jobs before t_s, and the other part is the demand of a carry-over job J_H. In this case, the probabilistic processor demand is equal to $\mathcal{DBF}(2) = (\mathcal{V}(k_i) \odot C_i^{LO}) \bigotimes \mathcal{DBF}(J_H, t)$.

In short, the probabilistic processor demand of any task $\tau_i \in \Gamma_{HI}$ within $[0, t)$ and $D_i \le t - t_s$ in HI mode can be determined as follows:

$$\mathcal{DBF}_H^H(\tau_i, t) = \begin{cases} \mathcal{DBF}(1), & D_i \le t - t_s; \\ \mathcal{M}, & Otherwise, \end{cases} \tag{15}$$

where $\mathcal{M}$ can be determined as follows:

$$\mathcal{M} = \begin{cases} \mathcal{DBF}(1), & \max\{\mathcal{DBF}(2)\} \le \max\{\mathcal{DBF}(1)\}; \\ \mathcal{DBF}(2), & Otherwise. \end{cases} \tag{16}$$

Therefore, the probabilistic processor demand of all tasks within $[0, t)$ in HI mode is determined by the following:

$$\mathcal{DBF}_H(t) = \Big(\bigotimes_{\tau_i \in \Gamma_{LO}} \mathcal{DBF}_L^H(\tau_i, t)\Big) \bigotimes \Big(\bigotimes_{\tau_i \in \Gamma_{HI}} \mathcal{DBF}_H^H(\tau_i, t)\Big). \tag{17}$$

Theorem 3. *An IMC taskset Γ is deterministically schedulable under EDF, if $0 < \forall t \le t_{max}$, $0 < t_s < t$,*

$$\max\{\mathcal{DBF}_L(t)\} \le t, \quad and \quad \max\{\mathcal{DBF}_H(t)\} \le t, \tag{18}$$

It is probabilistically schedulable if the maximum probability that the processor demand of all tasks in both LO mode and HI mode exceeds t does not exceed the permitted system failure probability F_s,[2] expressed as:

$$1 - \Pi_{t_k=t}^{t_{max}} F_{\mathcal{DBF}_L(t_k)}(t_k) \le F_s, and \tag{19}$$
$$1 - \Pi_{t_k=t}^{t_{max}} F_{\mathcal{DBF}_H(t_k)}(t_k) \le F_s.$$

[2] Chen et al. [38] pointed out that there are certain flaws in the probabilistic WCRT based on critical instant instances. However, our work focuses on the overall distribution of all task behaviors within a task's hyper-period, rather than relying solely on a single critical instant and considers the probability distribution of all possible processor demand throughout the hyper-period.

Table 3
Taskset parameters of Γ_2, with $C_1^{deg} = 3, C_2^{thr} = 1, C_3^{deg} = 3$.

Task	L_i	$T_i = D_i$	C_i^{LO}	C_i^{HI}
τ_1	LO	10	$\begin{pmatrix} 1 & 3 & 4 & 5 \\ 0.455 & 0.54 & 0.004 & 0.001 \\ 0.455 & 0.995 & 0.999 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1 & 3 \\ 0.455 & 0.545 \\ 0.455 & 1.0 \end{pmatrix}$
τ_2	HI	20	$\begin{pmatrix} 0.5 & 1 \\ 0.49 & 0.51 \\ 0.49 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 0.5 & 1 & 2 & 3 \\ 0.49 & 0.5 & 0.009 & 0.001 \\ 0.49 & 0.99 & 0.999 & 1.0 \end{pmatrix}$
τ_3	LO	10	$\begin{pmatrix} 2 & 3 & 4 & 5 \\ 0.019 & 0.6 & 0.38 & 0.001 \\ 0.019 & 0.619 & 0.999 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 2 & 3 \\ 0.019 & 0.981 \\ 0.019 & 1.0 \end{pmatrix}$

Proof. The IMC taskset Γ is *deterministically schedulable* under EDF if it is *deterministically schedulable* in both LO mode and HI mode. The condition for deterministic schedulability in LO mode and HI mode Eq. (18) is self-evident, because it can be directly derived from Theorems 1 and 2. In addition, the IMC taskset Γ is *probabilistically schedulable* under EDF if it is *probabilistically schedulable* in both LO mode and HI mode. The condition for probabilistic schedulability (Eq. (19)) states that the probability that the processor demand of all tasks in both LO mode and HI mode exceeds t is less than or equal to F_s, hence it is probabilistically schedulable with system failure probability not exceeding F_s. (Note that the condition of deterministic schedulability in Eq. (18) is a special case of the condition of probabilistic schedulability in Eq. (19), with permitted system failure probability equal to 0 ($F_s = 0$).) Q.E.D.

In the deterministic analysis, the processor demand grows in a stepwise manner based on the interval length. The processor demand is affected only when the increase in interval length is a multiple of the task period. When we switch to probabilistic analysis, the probability distribution of processor demand also increases in a stepwise manner to maintain consistency. In other words, during deterministic analysis, the processor demand does not change in the given time intervals, and in probabilistic scheduling analysis, the values in its probability distribution of processor demand also remain unchanged. Specifically, there are some t_k values that can generate the same probability distribution of processor demand. The values of $F_{DBF_{L(t_k)}}(t_k)$ and $F_{DBF_{H(t_k)}}(t_k)$, which correspond to the same probability distribution of processor demand, should not be computed repeatedly in Eq. (19). Therefore, we only calculate once. In addition, If t_1, t_2 and t_l ($t_1 < t_2 < t_l$) can generate the same probability distribution of the processor demand for all tasks in both modes. We choose the minimum value t_1 among these values, which corresponds to $F_{DBF_{L(t_1)}}(t_1)$ and $F_{DBF_{H(t_1)}}(t_1)$. This is because it is the value that maximizes the probability of the processor demand exceeding the interval length.

4.2. Example 2

We present a taskset Γ_2, with the parameters shown in Table 3. (The nominal pWCET C_i is omitted for brevity.) We assume that $F_s = 1.0 \times 10^{-6}$.

In this example, $t_{max} = 20$. $0 < t < 10$, $0 < t_s < t$, we have $m_i = -1$ ($m_i = \lfloor \frac{t-D_i}{T_i} \rfloor$), $k_i = 0$ ($k_i = \lfloor \frac{t_s}{T_i} \rfloor$), $a_i = 0$, $c_i = 0$, and $b_i = 0$ ($i = 1, 2, 3$). According to Eq. (10), $DBF_L(\tau_i, t) = \mathcal{V}(0)$. In addition, we have $DBF_L(t) = \mathcal{V}(0)$ from Eq. (11). From Eq. (12), we have $DBF(J_L, t) = \mathcal{V}(0)$ for LO tasks τ_1 and τ_3. Moreover, we have $DBF(J_H, t) = \mathcal{V}(0)$ for HI task τ_2 from Eq. (13). Therefore, we have $DBF_L^H(\tau_1, t) = \mathcal{V}(0)$ and $DBF_L^H(\tau_3, t) = \mathcal{V}(0)$ from Eq. (14). Due to $k_2 = 0$, $a_2 = 0$, $b_2 = 0$ and $D_2 > t - t_s$, we have $DBF(1) = \mathcal{V}(0)$, $DBF(2) = \mathcal{V}(0)$, and $\max\{DBF(2)\} \leq \max\{DBF(1)\}$. According to Eq. (15), we have $DBF_H^H(\tau_2, t) = \mathcal{V}(0)$. We calculate $DBF_H(t) = \mathcal{V}(0)$ from Eq. (17). Therefore, we have $\max\{DBF_L(t)\} \leq t$ and $\max\{DBF_H(t) \leq t\}$.

When $t = 10$, $m_1 = 0$, $m_2 = -1$, $m_3 = 0$, $k_i = 0$, $a_i = 0$, $c_i = 0$, and $b_i = 0$ ($i = 1, 2, 3$). According to Eq. (10), we have $DBF_L(\tau_1, t) = C_1^{LO}$, $DBF_L(\tau_2, t) = \mathcal{V}(0)$, and $DBF_L(\tau_3, t) = C_3^{LO}$. In addition, we have

$$DBF_L(t) = \mathcal{L} = \begin{pmatrix} 3 & 4 & \cdots & 8 & 9 & 10 \\ 0.008645 & 0.273 & \cdots & 0.00266 & 0.000384 & 0.000001 \\ 0.008645 & 0.281645 & \cdots & 0.999615 & 0.999999 & 1.0 \end{pmatrix}$$

from Eq. (11). Moreover, from (17), we have $DBF_H(t) = \mathcal{L}$.

When $10 < t < 20$, $m_1 = 0$, $m_2 = -1$, $m_3 = 0$, $a_i = 0$, $c_i = 0$, and $b_i = 0$ ($i = 1, 2, 3$). According to Eq. (11), we have $DBF_L(t) = \mathcal{L}$. If $t_s < 10$, $k_i = 0$ ($i = 1, 2, 3$). According to Eq. (17), we have $DBF_H(t) = \mathcal{L}$ and $\max\{DBF_H(t) \leq t\}$. If $10 \leq t_s < t$, we have $k_1 = 1$, $k_2 = 0$, and $k_3 = 1$. According to Eq. (14), we have $DBF_L^H(\tau_1, t) = C_1^{LO}$ and $DBF_L^H(\tau_3, t) = C_3^{LO}$. We calculate $DBF_H^H(\tau_2, t) = \mathcal{V}(0)$ from Eq. (15). In addition, we have $DBF_H(t) = \mathcal{L}$ from Eq. (17). Therefore, we have $\max\{DBF_H(t)\} \leq t$ and $\max\{DBF_L(t)\} \leq t$.

When $t = 20$, $m_1 = 1$, $m_2 = 0$, $m_3 = 1$. According to Eq. (10), we have $DBF_L(\tau_1, t) = \mathcal{V}(2) \odot C_1^{LO}$, $DBF_L(\tau_2, t) = C_2^{LO}$, and $DBF_L(\tau_3, t) = \mathcal{V}(2) \odot C_3^{LO}$. In addition, we have

$$DBF_L(t) = \begin{pmatrix} 6.5 & \cdots & 19 & 20.5 & 21 \\ 0.00423605 & \cdots & 0.00019584 & 0.00000049 & 0.00000051 \\ 0.00406315 & \cdots & 0.999999 & 0.99999949 & 1.0 \end{pmatrix}$$

from Eq. (11). If $t_s < 10$, $a_1 = 1$, $a_2 = 0$, $a_3 = 1$, $c_1 = 1$, $c_2 = 0$, $c_3 = 1$, $k_i = 0$, and $b_i = 0$ ($i = 1, 2, 3$). From Eq. (17), we have $\max\{DBF_H(t)\} = 19$. If $10 \leq t_s < t$, $k_1 = 1$, $k_2 = 0$, $k_3 = 1$, $b_1 = 1$, $b_2 = 0$, $b_3 = 1$, $a_i = 0$ and $c_i = 0$ ($i = 1, 2, 3$). According to Eq. (17), we have $\max\{DBF_H(t)\} = 23$. Therefore, we have $\max\{DBF_L(t)\} > t$ and $\max\{DBF_H(t)\} > t$ ($10 \leq t_s < t$), but $1 - F_{DBF_{L(t)}}(t) \leq F_s$ and $1 - F_{DBF_{H(t)}}(t) \leq F_s$. According to Theorem 3, the taskset Γ is *probabilistically schedulable*.

5. Energy-efficient task execution model

We present in sequence the power model, the calculation of energy-efficient processor speeds in LO mode, and the Energy-Efficient Task Execution Model in this section.

5.1. Power model

We adopt the state-of-the-art processor power model [39–41]

$$P = P_s + h(P_{ind} + C_{ef} s^m), \tag{20}$$

where P_s is a static power and P_{ind} is the frequency-independent active power. $h = 1$ if the system is active (defined as having computation in progress); otherwise, $h = 0$. C_{ef} is an effective switching capacitance and m is system-application-dependent constant. s is the normalized processor speed (frequency). Like [39], we ignore a static power ($P_s = 0$) and set $P_{ind} = 0.01, C_{ef} = 1, m = 3$.

Considering our task model, the expected energy consumption of a single job of task τ_i is [42–44]:

$$\overline{E}_i = (P_{ind} + C_{ef} s^m) \cdot \frac{\overline{x}_i}{s} \tag{21}$$

where $\overline{x}_i = \sum_{k=0}^{K-1} C_i^k \cdot f_{C_i^{LO}}(C_i^k)$ with the normalized processor speed $S_{max} = 1$. In addition, the processor speed s should not be lower than S_{crit}, where S_{crit} ($S_{crit} < S_{max}$) is an energy-efficient speed while it can be computed $S_{crit} = \sqrt[m]{\frac{P_{ind}}{(m-1) \cdot C_{ef}}}$ [39].

To facilitate comparisons between task sets with varying hyper-periods, we utilize the definition of *normalized energy consumption* of task set Γ within its hyper-period [22] (i.e., its power consumption):

$$\begin{aligned} \overline{NE}(\Gamma) &= \frac{1}{HP(\Gamma)} \sum_{i=1}^{n} \sum_{j=1}^{h_i} (P_{ind} + C_{ef} s^m) \cdot \frac{\overline{x}_i}{s} \\ &= \sum_{i=1}^{n} (P_{ind} + C_{ef} s^m) \cdot \frac{\overline{x}_i}{s \cdot T_i}, \end{aligned} \tag{22}$$

where $h_i = HP(\Gamma)/T_i$ is the number of jobs of task $\tau_i \in \Gamma$ released in the hyper-period $HP(\Gamma)$.

5.2. Calculating energy-efficient processor speeds

We determine the energy-efficient processor speed in LO mode S_L and schedule the tasks with $S_{max} = 1$ in HI mode if an IMC taskset Γ is deterministically schedulable by EDF on a single processor.

A taskset Γ running on a processor with speed S_L is equivalent to the taskset Γ^* running on a processor with speed $S_{\max} = 1$ with proportionally-scaled execution times $1/S_L$ times of each task in Γ. Therefore, the probabilistic processor demand of any task $\tau_i \in \Gamma$ with speed S_L within $[0, t)$ in LO mode can be calculated as follows:

$$\mathcal{DBF}_L(\tau_i, t) = \mathcal{V}(\llbracket m_i + 1 \rrbracket_0) \odot (\mathcal{V}(1/S_L) \odot \mathcal{C}_i^{LO}), \tag{23}$$

The probabilistic processor demand of a carry-over job released by LO task τ_i with speed S_L within $[0, t)$ can be calculated as follows:

$$\mathcal{DBF}(J_L, t) = \begin{cases} \mathcal{V}(1/S_L) \odot \mathcal{C}_i^{LO}, & r_i + D_i \le t \\ \mathcal{V}(0), & otherwise. \end{cases} \tag{24}$$

The probabilistic processor demand of any task $\tau_i \in \Gamma_{LO}$ with speed S_L within $[0, t)$ in HI mode can be calculated as follows:

$$\begin{aligned} \mathcal{DBF}_L^H(\tau_i, t) =& (\mathcal{V}(k_i) \odot (\mathcal{V}(1/S_L) \odot \mathcal{C}_i^{LO})) \bigotimes \\ & \mathcal{DBF}(J_L, t) \bigotimes (\mathcal{V}(c_i) \odot \mathcal{C}_i^{HI}). \end{aligned} \tag{25}$$

In addition, the system schedules tasks with S_L in LO mode and $S_{max} = 1$ in HI mode, $\mathcal{DBF}(1)$ and $\mathcal{DBF}(2)$ in Eq. (16) are calculated by Eqs. (26) and (27), respectively.

$$\begin{aligned} \mathcal{DBF}(1) =& (\mathcal{V}(b_i) \odot (\mathcal{V}(1/S_L) \odot \mathcal{C}_i^{LO})) \bigotimes \\ & \mathcal{DBF}(J_H, t) \bigotimes (\mathcal{V}(a_i) \odot \mathcal{C}_i^{HI}). \end{aligned} \tag{26}$$

$$\mathcal{DBF}(2) = (\mathcal{V}(k_i) \odot (\mathcal{V}(1/S_L) \odot \mathcal{C}_i^{LO})) \bigotimes \mathcal{DBF}(J_H, t). \tag{27}$$

Theorem 4. *Given an IMC taskset Γ that is deterministically schedulable by EDF on a single processor, it remains deterministically schedulable with the energy-efficient processor speed S_L in LO mode and $S_{max} = 1$ in HI mode if $0 < \forall t \le t_{max}$, $0 < t_s < t$*

$$\max\{\mathcal{DBF}_L(t)\} \le t, \quad and \quad \max\{\mathcal{DBF}_H(t)\} \le t, \tag{28}$$

where $S_{crit} \le S_L \le 1$, $\mathcal{DBF}_L(\tau_i, t)$, $\mathcal{DBF}(J_L, t)$, $\mathcal{DBF}_L^H(\tau_i, t)$, $\mathcal{DBF}(1)$ and $\mathcal{DBF}(2)$ are given in Eqs. (23)–(27), respectively.

Proof. Theorem 4 can be directly derived from Theorem 3.

5.3. Example 3

Let us consider the task set Γ_3 that consists of tasks with the parameters presented in Table 4. The processor has tens discrete normalized processor speed, i.e., $[0.1, 0.2, \ldots, 1.0]$ [45]. According to Theorem 3, the taskset is deterministically schedulable in both modes. We calculate $S_L = 0.8$ on the basis of Theorem 4, by iteratively trying out the available speeds, from lowest to highest, until we find the minimum speed that satisfies all constraints. According to Eq. (21), we have $\bar{x}_1 = 1.775$, $\bar{x}_2 = 1.99$, $\bar{x}_3 = 2.2$. In addition, we can then use Eq. (22) to obtain the taskset's normalized energy consumption to be 0.3242925 with processor speed $S_L = 0.8$ with DVFS, and 0.50197 with processor speed $S_{\max} = 1$ for EDF without DVFS, which represents significant energy savings.

5.4. Energy-efficient task execution model

Assuming that the system is deterministically schedulable in both modes, we can use DVFS to reduce the processor speed to S_L in LO mode, and set to $S_{max} = 1$ in HI mode, while maintaining schedulability in both modes. We modify the task execution model in Section 3.1 to be the *energy-efficient task execution model* based on DVFS as shown below.

Table 4
Taskset parameters of Γ_3, with $C_1^{deg} = 1.5, C_2^{thr} = 2, C_3^{deg} = 2$.

Task	L_i	$T_i = D_i$	$\mathcal{C}_i^{LO}$	$\mathcal{C}_i^{HI}$
τ_1	LO	10	$\begin{pmatrix} 1 & 1.5 & 2 & 2.5 \\ 0.1 & 0.4 & 0.35 & 0.15 \\ 0.1 & 0.5 & 0.85 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1.5 \\ 0.1 & 0.9 \\ 0.1 & 1.0 \end{pmatrix}$
τ_2	HI	20	$\begin{pmatrix} 1 & 2 \\ 0.01 & 0.99 \\ 0.01 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 & 4 & 5 \\ 0.01 & 0.49 & 0.45 & 0.05 \\ 0.01 & 0.5 & 0.95 & 1.0 \end{pmatrix}$
τ_3	LO	10	$\begin{pmatrix} 1.5 & 2 & 2.5 & 3 \\ 0.2 & 0.3 & 0.4 & 0.1 \\ 0.2 & 0.5 & 0.9 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1.5 & 2 \\ 0.2 & 0.8 \\ 0.2 & 1.0 \end{pmatrix}$

Energy-efficient task execution model in probabilistic IMC. The system is first initialized to be in LO mode with processor speed S_L. If any HI task $\tau_i \in \Gamma_{HI}$ executes beyond its C_i^{thr}/S_L, the system switches into HI mode, with processor speed $S_{max} = 1$. As the mode-switch instant, if jobs of LO tasks have run for longer than their C_i^{deg}/S_L, the jobs will be stopped until new released. In addition, if the execution time of LO jobs is less than C_i^{deg}/S_L by the switch time instant, these carry-over jobs will continue to execute the leftover execution up to $C_i^{leftover}$ after the switch time instant and before their deadlines, where $C_i^{leftover}$ is the leftover execution time at the nominal processor speed $S_{max} = 1$. While in HI mode, each LO task $\tau_i \in \Gamma_{LO}$ executes no more than its C_i^{deg} if it is started in HI mode, or its $C_i^{leftover}$ if it is a leftover job started in LO mode. The system switches back to LO mode, with processor speed S_L, at an idle instant if no jobs wait for executions at this time. In addition, incomplete tasks are dropped at their deadlines, hence there does not exist a backlog of outstanding execution at the end of each hyper-period.

6. Experimental evaluation

We evaluate our approach based on two performance metrics: the *schedulability ratio*, which represents the proportion of schedulable task sets (either deterministically or probabilistically schedulable) out of all task sets; and the *normalized energy consumption* of each task set, as defined in Eq. (22).

We generate synthetic tasksets based on the following experiment settings:

- Number of tasks in each taskset Γ is set to $n = 4$.
- Number of HI tasks in Γ is set to $n \cdot CP$, where the Criticality Proportion CP is set to $CP = 0.5$.
- Number of discrete values of each task τ_i's nominal pWCET C_i is set to $K = 4$.
- Each of the K probability values in the PMF of C_i is selected randomly from $[0, 1)$ while ensuring that they sum to 1 (similar to [46,47]).
- For each LO task $\tau_i \in \Gamma_{LO}$, the index of the Degraded WCET C_i^{deg} among the K discrete values of C_i is set to $ind(C_i^{deg}) = 0.5K - 1 = 1$.
- For each HI task $\tau_i \in \Gamma_{HI}$, the index of the Threshold WCET C_i^{thr} among the K discrete values of C_i is set to $ind(C_i^{thr}) = 0.5K - 1 = 1$.
- T_i is randomly selected the set $\{10, 20, 40, 50, 100, 200, 400, 500, 1000\}$ [48].
- To control taskset processor utilization, $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$ is varied from 0.1 to 0.9, in steps of 0.1, while $\max\{\mathcal{U}_{HI}^{HI}(\Gamma)\}$ is chosen randomly from the range $[0.1, 1.0]$.

(Each task τ_i's pWCET C_i and period T_i are implicit, since both system schedulability and normalized energy consumption are dependent on the utilization values only, i.e., pWCU equal to pWCET divided by period.) Note that the time overhead of the proposed method is mainly

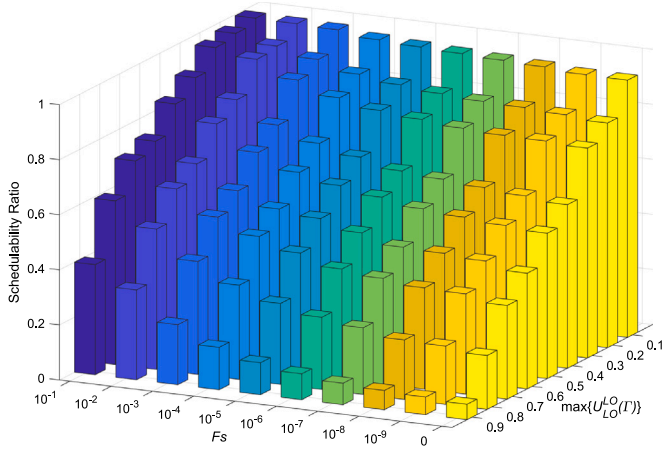


Fig. 2. Impact on the schedulability ratio by varying the permitted system failure probability F_s and $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$.

spent on the schedulability test, with significant time consumption arising from the calculation of the probabilistic processor demands for the task set, which involves a large number of convolution operations. As the number of tasks increases, the time overhead grows exponentially. To maintain the accuracy of the scheduling test, we have not yet identified better methods to reduce the time overhead. Hence, we have limited the number of tasks to four. In the future, we will strive to reduce the time overhead associated with convolutions.

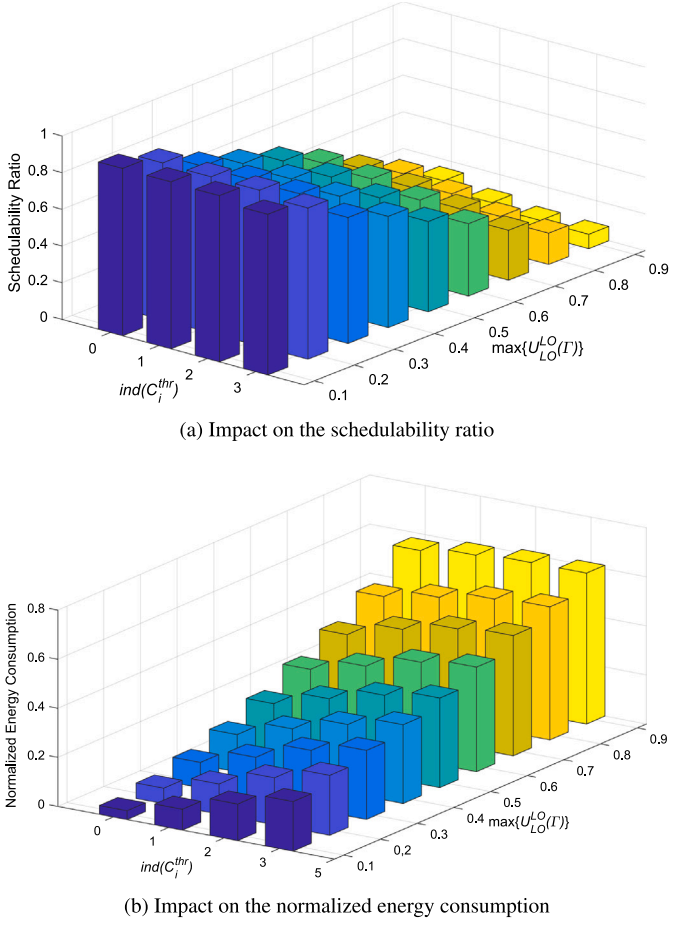
In the first experiment, we vary F_s from 10^{-1} to 10^{-9} with a step size of 10 by multiplication, i.e., F_s is plotted with log scale. The value $F_s = 10^{-9}$ is based on the permitted failure probability of 10^{-9} for ASIL D, the highest safety certification level in ISO 26262. The additional case of $F_s = 0$ is the special case of deterministic schedulability only for hard real-time systems. Fig. 2 shows the results, where each data point represents the average outcome obtained from a variable number of task sets selected from 500 synthetic tasksets generated for each value of $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$, using different seeds for the pseudo-random number generator.

We make the following observations from Fig. 2:

- The schedulability ratio is positively correlated with F_s, confirming the significant advantages of considering probabilistic schedulability compared to considering deterministic schedulability only, even at very small values of F_s for high levels of safety certification.
- The schedulability ratio is negatively correlated with $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$, since both $\max\{\mathcal{DBF}_L(t)\}$ and $\max\{\mathcal{DBF}_H(t)\}$ increase with increasing $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$, which reduces system schedulability.

In the second experiment, we fix the permitted system failure probability to be $F_s = 10^{-7}$ (based on the requirement for ASIL A in ISO 26262). We vary each HI task's C_i^{thr} through varying its index $ind(C_i^{thr})$ from 0 to $K-1$ with step size 1, i.e., the sequence $\{0,1,2,3\}$ (The case of $ind(C_i^{thr}) = 3$ is the special case where each HI task τ_i has the same WCET in both modes.). Each LO task's C_i^{deg} is fixed to be the default value of $ind(C_i^{deg}) = 1$. The results are shown in Fig. 3, including both the schedulability ratio, and the normalized energy consumption ($\overline{NE}(\Gamma)$ defined in Eq. (22)). Each data point represents the average outcome obtained from a variable number of task sets selected from 500 synthetic tasksets generated for each value of $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$, depending on the value of $ind(C_i^{thr})$.

We make the following observations from Fig. 3:



(a) Impact on the schedulability ratio

(b) Impact on the normalized energy consumption

Fig. 3. Varying each HI task's Threshold WCET index $ind(C_i^{thr})$ and $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$.

- The schedulability ratio is negatively correlated with $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$, as expected.
- The schedulability ratio is negatively correlated with C_i^{thr}. With increasing C_i^{thr}, HI tasks have larger WCETs (both expected and maximum) in LO mode according to the trimming operation for pWCET defined in Eq. (2), causing $\max\{\mathcal{DBF}_L(t)\}$ and $\max\{\mathcal{DBF}_H(t)\}$ to increase, which reduces system schedulability.
- The average normalized energy consumption $\overline{NE}(\Gamma)$ is positively correlated with $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$. From Eq. (22), $\overline{NE}(\Gamma)$ is dependent on each task's expected pWCET $\overline{x}_i$ and the energy-efficient processor speed in LO mode S_L. With increasing $\max\{\mathcal{U}_{LO}^{LO}(\Gamma)\}$, both $\overline{x}_i$ and S_L increase, causing $\overline{NE}(\Gamma)$ to increase.
- $\overline{NE}(\Gamma)$ is positively correlated with C_i^{thr}. With increasing C_i^{thr}, HI task τ_i has a larger expected pWCET in LO mode, causing both $\overline{x}_i$ and S_L to increase, which in turn causes $\overline{NE}(\Gamma)$ to increase.

Averaged over all cases, our approach achieves an average reduction of 33.49% for the average normalized energy consumption compared to EDF without DVFS.

7. Practical considerations

In this section, we address some practical considerations in transposing our proposal into to industry practice.

Timing analysis for pWCET. Task τ_i's pWCET C_i, as specified by its PMF, may be obtained via static, dynamic or measurement-based, or hybrid timing analysis methods, as discussed in the survey paper [49]. Static Probabilistic Timing Analysis (SPTA) is based on the analysis of the program code, along with an abstract model of the

hardware behavior. Measurement-Based Probabilistic Timing Analysis (MBPTA) typically applies Extreme Value Theory (EVT) to make a statistical estimate of the pWCET distribution of a program. Hybrid Probabilistic Timing Analysis (HyPTA) combines both statistical and analytical approaches, e.g., by taking measurements at the level of basic blocks or sub-paths, and then composing the results using structural information obtained from static analysis of the code.

Number of discrete value (K) of pWCET C_i. The value of K determines the granularity of modeling the pWCET's PMF: larger K implies finer granularity modeling, but may not be well-supported by timing analysis techniques, and also leads to higher computational costs in schedulability analysis. The typical value of K is 2-8 [5], although there is no hard lower or upper bound on its value. Our experiments with K varying from 4 to 8 indicate that its value does not affect system schedulability and power consumption significantly, indicating that $K = 4$ already provides sufficiently fine granularity modeling under our experimental setup.

PMF of pWCET C_i. In the absence of real industry tasksets, we need to generate each task's pWCET C_i synthetically, as defined by the PMF. There is no clear consensus on the generation method in the literature on probabilistic schedulability analysis. An early work Edgar and Burns [50] used the trimmed and scaled Gumbel distribution to model likely WCET values; Draskovic [36] used the Weibull distribution with an upper bound, which was used for modeling the distribution of long but unlikely execution times based on EVT [51] (the Log of a Weibull distribution is a Gumbel distribution); Wang et al. [46] and Markovic et al. [47] adopted the uniform random distribution; Bozhko et al. [52] assumed two execution modes for each task in an MCS: a typical mode and a rare exceptional mode. Its pWCET is equal to c with probability .95 (the typical mode), and $4c$ with probability .05 (the exceptional mode), where c was scaled to match the expected task utilization. In this paper, we adopt the simple approach of the uniform random distribution similar to [46,47].

Runtime overhead of DVFS. The overhead of varying the processor speed with DVFS is assumed to be zero. This is a common assumption adopted in the DVFS literature [7]. We can determine through offline measurement an upper bound on the processor speed transition overhead, which is typically relatively small compared to the WCET of the task, hence it can be added to each task's execution time without a significant impact on the solution.

Multiprocessor platforms. Our work can be easily extended to multi-processor platforms by a partitioned scheduling approach [31,32,53]. In partitioned scheduling, tasks are statically assigned to processors, with each processor managed by a local scheduler. We can use simple allocation methods, e.g., Criticality-unaware worst-fit decreasing (CU-WFD), and criticality-aware first-fit decreasing (CA-FFD), to allocate tasks to each processor while using an Energy-Efficient Task Execution Model to schedule tasks in each processor.

8. Conclusions and future work

The classic MCS task model has several restrictive assumptions, including hard real-time constraints, dropping LO tasks in HI mode, and lack of consideration of power/energy consumption issues. In this paper, we relax these assumptions to make the MCS task model more practically applicable. We consider an IMC taskset scheduled with the EDF algorithm on a uniprocessor platform, and propose an Energy-Efficient Task Execution Model that guarantees (deterministic or probabilistic) schedulability, allows degraded QoS to LO tasks in HI mode, and applies DVFS to save energy.

In this paper, we have considered EDF-based uniprocessor scheduling, dual-criticality MCS, and task execution time as probabilistic variables. As part of future work, these assumptions can be further relaxed to fixed-priority scheduling, multi-processor platforms, multiple criticality levels, and the multiple task parameters (e.g., task period) represented by random variables.

CRediT authorship contribution statement

Yi-Wen Zhang: Writing – review & editing, Writing – original draft, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Jin-Long Zhang:** Writing – original draft, Visualization, Software, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the Natural Science Foundation of Fujian Province of China under Grant 2023J01139 and the Fundamental Research Funds for the Central Universities, China under Grant ZQN-1009.

Data availability

No data was used for the research described in the article.

References

[1] Alan Burns, Robert Ian Davis, Mixed criticality systems-a review:(february 2022), 2022, pp. 1–97, https://eprints.whiterose.ac.uk/183619/.
[2] Steve Vestal, Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance, in: 28th IEEE International Real-Time Systems Symposium, RTSS 2007, IEEE, 2007, pp. 239–243.
[3] Yi-Wen Zhang, Jin-Peng Ma, Hui Zheng, Zonghua Gu, Criticality-aware EDF scheduling for constrained-deadline imprecise mixed-criticality systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 43 (2) (2024) 480–491.
[4] Yi-Wen Zhang, Hui Zheng, Slack time management for imprecise mixed-criticality systems with reliability constraints, IEEE Trans. Comput. (2025).
[5] Robert I. Davis, Liliana Cucu-Grosjean, A survey of probabilistic schedulability analysis techniques for real-time systems, Leibniz Trans. Embed. Syst. 6 (1) (2019) 04:1–04:53.
[6] Yi-Wen Zhang, Rong-Kun Chen, A survey of energy-aware scheduling in mixed-criticality systems, J. Syst. Archit. 127 (2022) 102524.
[7] Ashikahmed Bhuiyan, Federico Reghenzani, William Fornaciari, Zhishan Guo, Optimizing energy in non-preemptive mixed-criticality scheduling by exploiting probabilistic information, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (11) (2020) 3906–3917.
[8] Yi-Wen Zhang, Chen Ouyang, Semi-clairvoyant scheduling in non-preemptive fixed-priority mixed-criticality systems, J. Syst. Archit. 159 (2025) 103332.
[9] Qingling Zhao, Mengfei Qu, Zonghua Gu, Haibo Zeng, Minimizing stack memory for partitioned mixed-criticality scheduling on multiprocessor platforms, ACM Trans. Embed. Comput. Syst. (TECS) 21 (2) (2022) 1–30.
[10] Benny Akesson, Mitra Nasri, Geoffrey Nelissen, Sebastian Altmeyer, Robert I Davis, A comprehensive survey of industry practice in real-time systems, Real-Time Syst. (2021) 1–41.
[11] Georg von der Brüggen, Nico Piatkowski, Kuan-Hsun Chen, Jian-Jia Chen, Katharina Morik, Björn B Brandenburg, Efficiently approximating the worst-case deadline failure probability under EDF, in: 2021 IEEE Real-Time Systems Symposium, RTSS, IEEE, 2021, pp. 214–226.
[12] Alexandre Esper, Geoffrey Nelissen, Vincent Nélis, Eduardo Tovar, An industrial view on the common academic understanding of mixed-criticality systems, Real-Time Syst. 54 (3) (2018) 745–795.
[13] Sanjoy Baruah, Alan Burns, Implementing mixed criticality systems in ADA, in: International Conference on Reliable Software Technologies, Springer, 2011, pp. 174–188.
[14] Sanjoy K. Baruah, Alan Burns, Robert I. Davis, Response-time analysis for mixed criticality systems, in: 2011 IEEE 32nd Real-Time Systems Symposium, IEEE Computer Society, 2011, pp. 34–43.
[15] François Santy, Gurulingesh Raravi, Geoffrey Nelissen, Vincent Nelis, Pratyush Kumar, Joël Goossens, Eduardo Tovar, Two protocols to reduce the criticality level of multiprocessor mixed-criticality systems, in: Proceedings of the 21st International Conference on Real-Time Networks and Systems, 2013, pp. 183–192.

[16] Sanjoy Baruah, Vincenzo Bonifaci, Gianlorenzo DAngelo, Haohan Li, Alberto Marchetti-Spaccamela, Suzanne Van Der Ster, Leen Stougie, The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems, in: 2012 24th Euromicro Conference on Real-Time Systems, IEEE, 2012, pp. 145–154.
[17] Di Liu, Nan Guan, Jelena Spasic, Gang Chen, Songran Liu, Todor Stefanov, Wang Yi, Scheduling analysis of imprecise mixed-criticality real-time tasks, IEEE Trans. Comput. 67 (7) (2018) 975–991.
[18] Hang Su, Nan Guan, Dakai Zhu, Service guarantee exploration for mixed-criticality systems, in: 2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications, IEEE, 2014, pp. 1–10.
[19] Robert I. Davis, Alan Burns, Iain Bate, Compensating adaptive mixed criticality scheduling, in: Proceedings of the 30th International Conference on Real-Time Networks and Systems, Association for Computing Machinery, 2022, pp. 81–93.
[20] Zhe Jiang, Xiaotian Dai, Alan Burns, Neil Audsley, Zonghua Gu, Ian Gray, A high-resilience imprecise computing architecture for mixed-criticality systems, IEEE Trans. Comput. (2022).
[21] Yi-Wen Zhang, Rui-Feng Guo, Low-power scheduling algorithms for sporadic task with shared resources in hard real-time systems, Comput. J. 58 (7) (2015) 1585–1597.
[22] Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, Lothar Thiele, Energy efficient dvfs scheduling for mixed-criticality systems, in: 2014 International Conference on Embedded Software, EMSOFT, IEEE, 2014, pp. 1–10.
[23] Yi-Wen Zhang, Energy-aware mixed-criticality sporadic task scheduling algorithm, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (1) (2021) 78–86.
[24] Yi-Wen Zhang, Rong-Kun Chen, Energy aware fixed priority scheduling in mixed-criticality systems, Comput. Stand. Interfaces 83 (2023) 103671.
[25] Yi-Wen Zhang, Energy efficient non-preemptive scheduling of imprecise mixed-criticality real-time tasks, Sustain. Comput.: Inform. Syst. 37 (2023) 100840.
[26] Yi-Wen Zhang, Ning Cai, Energy efficient EDF-VD-based mixed-criticality scheduling with shared resources, J. Syst. Archit. 119 (2021) 102246.
[27] Y.-W. Zhang, Energy aware algorithm based on actual utilization for periodic tasks in mixed-criticality real-time systems, Comput. Stand. Interfaces 79 (2022) 103563.
[28] Yi-Wen Zhang, DVFS-based energy-aware scheduling of imprecise mixed-criticality real-time tasks, J. Syst. Archit. 137 (2023) 102849.
[29] Sujay Narayana, Pengcheng Huang, Georgia Giannopoulou, Lothar Thiele, R Venkatesha Prasad, Exploring energy saving for mixed-criticality systems on multi-cores, in: 2016 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS, IEEE, 2016, pp. 1–12.
[30] Behnaz Ranjbar, Tuan D.A. Nguyen, Alireza Ejlali, Akash Kumar, Power-aware runtime scheduler for mixed-criticality systems on multicore platform, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (10) (2021) 2009–2023.
[31] Yi-Wen Zhang, Rong-Kun Chen, Zonghua Gu, Energy-aware partitioned scheduling of imprecise mixed-criticality systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 42 (11) (2023) 3733–3742.
[32] Yi-Wen Zhang, Jin-Peng Ma, Zonghua Gu, Partitioned scheduling with shared resources on imprecise mixed-criticality multiprocessor systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 44 (1) (2025) 65–76.
[33] Luca Santinelli, Laurent George, Probabilities and mixed-criticalities: the probabilistic c-space, in: Proceedings of WMC, 2015.
[34] Dorin Maxim, Robert I Davis, Liliana Cucu-Grosjean, Arvind Easwaran, Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling, in: Proceedings of the 25th International Conference on Real-Time Networks and Systems, 2017, pp. 237–246.
[35] Jasdeep Singh, Luca Santinelli, Federico Reghenzani, Konstantinos Bletsas, Zhishan Guo, Non-preemptive scheduling of periodic mixed-criticality real-time systems, in: Proceedings of the 10th European Congress on Embedded Real-Time Systems, ERTS 2020, IEEE, 2020.
[36] Stefan Draskovic, Rehan Ahmed, Pengcheng Huang, Lothar Thiele, Schedulability of probabilistic mixed-criticality systems, Real-Time Syst. 57 (4) (2021) 397–442.
[37] Zhishan Guo, Sudharsan Vaidhun, Luca Satinelli, Samsil Arefin, Jun Wang, Kecheng Yang, Mixed-criticality scheduling upon permitted failure probability and dynamic priority, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 41 (1) (2021) 62–75.
[38] Kuan-Hsun Chen, Mario Günzel, Georg von der Brüggen, Jian-Jia Chen, Critical instant for probabilistic timing guarantees: Refuted and revisited, in: 2022 IEEE Real-Time Systems Symposium, RTSS, IEEE, 2022, pp. 145–157.
[39] Yifeng Guo, Dakai Zhu, Hakan Aydin, Jian-Jun Han, Laurence T Yang, Exploiting primary/backup mechanism for energy efficiency in dependable real-time systems, J. Syst. Archit. 78 (2017) 68–80.
[40] Yi-Wen Zhang, System level fixed priority energy management algorithm for embedded real time application, Microprocess. Microsyst. 64 (2019) 170–177.
[41] Yi-Wen Zhang, Chu-Gui Xu, Low power fixed priority scheduling sporadic task with shared resources in hard real time systems, Microprocess. Microsyst. 45 (2016) 164–175.
[42] Wei Jiang, Xiong Pan, Ke Jiang, Liang Wen, Qi Dong, Energy-aware design of stochastic applications with statistical deadline and reliability guarantees, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 38 (8) (2019) 1413–1426.
[43] Yi-Wen Zhang, Hui Zheng, Energy-aware fault-tolerant scheduling for imprecise mixed-criticality systems with semi-clairvoyance, J. Syst. Archit. 151 (2024) 103141.
[44] Yi-Wen Zhang, Hui Zheng, Energy-aware reliability guarantee scheduling with semi-clairvoyant in mixed-criticality systems, J. Syst. Archit. 156 (2024) 103269.
[45] Baoxian Zhao, Hakan Aydin, Dakai Zhu, Energy management under general task-level reliability constraints, in: 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium, IEEE, 2012, pp. 285–294.
[46] Tianyi Wang, Soamar Homsi, Linwei Niu, Shaolei Ren, Ou Bai, Gang Quan, Meikang Qiu, Harmonicity-aware task partitioning for fixed priority scheduling of probabilistic real-time tasks on multi-core platforms, ACM Trans. Embed. Comput. Syst. (TECS) 16 (4) (2017) 1–21.
[47] Filip Markovic, Thomas Nolte, Alessandro Vittorio Papadopoulos, Analytical approximations in probabilistic analysis of real-time systems, in: Proceedings of the 43rd IEEE Real-Time Systems Symposium, RTSS, IEEE, 2022.
[48] Jonah Caplan, Zaid Al-Bayati, Haibo Zeng, Brett H. Meyer, Mapping and scheduling mixed-criticality systems with on-demand redundancy, IEEE Trans. Comput. 67 (4) (2017) 582–588.
[49] Robert I. Davis, Liliana Cucu-Grosjean, A survey of probabilistic timing analysis techniques for real-time systems, LITES: Leibniz Trans. Embed. Syst. (2019) 1–60.
[50] Stewart Edgar, Alan Burns, Statistical analysis of WCET for scheduling, in: Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001)(Cat. No. 01PR1420), IEEE, 2001, pp. 215–224.
[51] Liliana Cucu-Grosjean, Luca Santinelli, Michael Houston, Code Lo, Tullio Vardanega, Leonidas Kosmidis, Jaume Abella, Enrico Mezzetti, Eduardo Quinones, Francisco J Cazorla, Measurement-based probabilistic timing analysis for multi-path programs, in: 2012 24th Euromicro Conference on Real-Time Systems, IEEE, 2012, pp. 91–101.
[52] Sergey Bozhko, Georg von der Brüggen, Björn Brandenburg, Monte carlo response-time analysis, in: IEEE 42nd Real-Time Systems Symposium, IEEE, 2021, pp. 342–355.
[53] Yi-Wen Zhang, Rong-Kun Chen, Energy-efficient scheduling of imprecise mixed-criticality real-time tasks based on genetic algorithm, J. Syst. Archit. 143 (2023) 102980.

Yi-Wen Zhang (Senior Member, IEEE) received his Ph.D in Computer Application Technology from University of Chinese Academy of Sciences in 2016. He was a Post-doctoral Fellow with Shenyang Institute of Computing Technology, Chinese Academy of Sciences from 2017 to 2019.

He has been an associate professor since 2020. He is named in the world's top 2% of Scientists List 2023 and 2024 by Stanford University. His current research interests include real-time systems and low-power design.

Jin-Long Zhang received the B.E. degree in Software Engineering from Jiangxi Agricultural University in 2023. He is currently pursuing the MS degree in Huaqiao University. His current research interests include real-time systems and low power design.