Energy-aware Mixed-criticality Sporadic Task Scheduling Algorithm

Yi-wen Zhang

***Abstract*—The mixed-criticality system provides multiple real-time applications with different criticalities in a single system. Poor energy saving performance of the previous studies on mixed-criticality sporadic tasks are mainly due to the fact that the slack time generated from the random arrival of sporadic tasks is not taken into account. In this paper, we focus on scheduling energy aware mixed-criticality sporadic tasks and take the random arrival of sporadic tasks into account. We proposed a dynamically frequency updating mixed-criticality algorithm (DFU). DFU based on earliest deadline first scheme can exploit the slack time generated from high criticality tasks in a low criticality mode to reduce processor frequency. In addition, it also can dynamically update the utilization of sporadic tasks set to further reduce processor frequency. The simulation experiments are conducted to evaluate the performance of DFU and experimental results show that DFU consumes 34.29% less energy than that of the existing algorithms.**

***Index Terms*—mixed-criticality system, sporadic task, energy-awareness, real-time scheduling**

I. INTRODUCTION

Mixed-criticality (MC) real-time system provides multiple real-time applications with different criticalities in a single system. It has been applied in many domains such as avionics, automotive and industrial standards (RTCA DO-178B) [4]. In such systems, tasks have different criticality levels. The high criticality level task must be completed correctly on time while losing a part of low criticality level task is tolerable.

Energy consumption is also very important in MC real-time systems. Actually, only a few recent studies focus on scheduling energy aware sporadic tasks in MC real-time systems. The authors in [16] first propose two speed levels ideas to reduce energy consumption in dynamic priority MC systems. But they only focus on the correction of the system. The authors in [8] first focus on scheduling energy aware sporadic tasks in MC real-time systems and propose a novel algorithm to reduce energy consumption. They expand the work given in [16] and first apply dynamic voltage and frequency scaling (DVFS) to adjust processor frequency. But they only exploit static slack time to reduce energy consumption, which leads to poor energy savings. The novel algorithm [9] not only exploits static slack time but also the reserved time for high criticality tasks to reduce energy consumption. But it ignores the random arrival of sporadic tasks, which can generate much slack time at run-time.

All in all, previous studies about scheduling energy aware sporadic tasks in MC real-time systems do not exploit the slack time generated from the random arrival of sporadic tasks. In this paper, we focus on scheduling energy aware MC sporadic tasks and take the random arrival of sporadic tasks into account. The main contributions are as follows:

1. A dynamically updating utilization algorithm is proposed and it can reclaim the slack time generated from the random arrival of sporadic tasks.
2. We proposed DFU based on earliest deadline first scheme, which not only exploits the slack time generated from the high criticality task in the low criticality mode, but also the slack time generated from random arrival of sporadic tasks.
3. We analyze scheduling feasibility of DFU.

The rest of this paper is organized as follows. We introduce the related work and the system model in Section II and Section III, respectively. We recap of earliest deadline first with virtual deadlines (EDF-VD) in Section IV. The proposed algorithm and simulation experiment are introduced in Section V and Section VI, respectively. Finally, we conclude with the summary in Section VII.

II. RELATED WORK

Many researchers focus on scheduling sporadic tasks in MC systems [1-2, 17-18]. The authors in [17] first address MC scheduling problem and then propose fixed priority scheduling schemes such as partitioned criticality (PC), static mixed criticality (SMC), and adaptive mixed criticality (AMC) [2]. In addition, the authors in [1] study the schedulability analysis for fixed priority sporadic tasks scheduling in MC systems and propose the own criticality based priority (OCBP)-schedulable method. Moreover, the generalizing fixed priority scheduling based on OCBP is proposed in [18]. Furthermore, the schedulability analysis to enable integration Preemption Threshold Scheduling with MC is proposed in [19]. Note that previous studies focus on fixed priority scheduling in MC systems. The authors in [15] focus on dynamic priority scheduling in MC systems and propose an Earliest Deadline First with Virtual Deadlines algorithm (EDF-VD).The MC sporadic task with multiple virtual deadlines based on EDF-VD is studied in [20]. In addition, a new demand-based schedulability test for general MC sporadic task systems and the new deadline tightening strategy based on this test are proposed in [27]. Furthermore, the flexible mixed-criticality

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, “This work was supported by the Natural Science Foundation of Fujian Province of china, Grant 2019J01080.”

The next few paragraphs should contain the authors’ current affiliations, including current address and e-mail. For example, Yi-wen Zhang is with the College of Computer Science and Technology, Huaqiao University, China, CO 361021, CHINA (e-mail:zyw@ hqu.edu.cn).

(FMC) model is proposed in [21] and the authors derive a utilization-based technique based on EDF-VD to analyze the schedulability of FMC model.

Note that the above studies focus on the schedulability analysis of MC systems and do not take energy consumption into account. DVFS is a general technology to reduce energy consumption. Many researchers [12, 22-25] apply DVFS to reduce energy consumption in traditional real-time systems. The event-triggered method is proposed to reclaim the slack time generated from sporadic tasks [22]. But it ignores the static power and does not exploit the slack time generated at run-time. The algorithm based on a slack time management queue is proposed in [12]. It not only exploits the slack time generated at run-time, but also considers the general power model. The above studies do not take shared resources into account. The problem of scheduling sporadic task with shared resources is addressed and a dynamic task synchronization algorithm is proposed in [23]. It combines DVFS and dynamic power management techniques to save energy. But it ignores the energy consumption of other components. The problem of system level energy consumption consisting of processor and other components is studied in [24].

Few studies focus on scheduling energy aware sporadic tasks in MC real-time systems. The first work in [8] studies the problem of scheduling energy aware sporadic tasks in MC real-time systems. The authors formulate a convex program by integrating DVFS with a well-known MC scheduling technique and propose an optimal algorithm. But they do not exploit the reserved time for high criticality tasks to reduce energy consumption. The authors in [9] extend the work in [8] and propose a novel algorithm. It not only reclaims static slack time, but also the reserved time for high criticality tasks to save energy. In addition, the authors in [30] focus on precise scheduling of all tasks of MC model and present schedulability tests based on utilization. Moreover, a minimum necessary execution speed is determined to reduce energy consumption. However, it does not exploit the dynamic slack time generated from the random arrival of sporadic tasks at run-time. In addition, the problem of scheduling energy aware sporadic tasks in MC system on chip (SoC) has been studies in [26]. Moreover, the problem of reliability in MC systems has been addressed in [4]. In short, poor energy saving performance of previous studies on MC sporadic tasks are mainly due to the fact that the slack time generated from the random arrival of sporadic tasks is not exploited to reduce energy consumption.

III. SYSTEM MODEL

A. Task Model

We consider a MC sporadic task set $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ which includes n independent sporadic tasks on a uniprocessor. Each MC sporadic task τ_i can be described by a tuple of parameters $(T_i, D_i, L_i, C_i(LO), C_i(HI))$, T_i and D_i are the minimum inter-arrival separation and a relative deadline of τ_i , respectively. $C_i(LO)$ and $C_i(HI)$ is the worst case execution time (WCET) of τ_i in a low criticality (LO) mode and in a high criticality (HI) mode, respectively. L_i is the criticality level set of τ_i (LO, HI, assuming a dual-criticality system). Each MC sporadic task can generate a finite number of jobs. Major notations can be found in Table 1. In this paper, we apply implicit-deadlines i.e. the relative deadline of τ_i is equal to T_i. If the criticality level of a task τ_i is equal to LO, we have $C_i(HI) = C_i(LO)$. If the criticality level of a task τ_i is equal to HI, we have $C_i(LO) \le C_i(HI)$ because the WCET in a HI mode is more conservative compared with WCET in a LO mode. In addition, we assume that the actual execution time of tasks is smaller or equal to its WCET in a LO mode. The utilization parameter is defined as follows:

$$U_y^z(\Gamma) = \sum_{\tau_i \in \Gamma \wedge L_i = y} \frac{C_i(z)}{T_i} \tag{1}$$

Where each of y and z in $\{LO, HI\}$ .Therefore, $U_{HI}^{LO}(\Gamma)$ means the sum of utilization of all HI tasks in Γ (i.e. the criticality level of a task is HI) in a LO mode.

System behavior. If a job of the task τ_i ends and its execution time is greater than $C_i(LO)$ and less than $C_i(HI)$, the system is in a HI mode. In addition, a job of the task τ_i ends and its execution time does not exceed $C_i(LO)$, the system is in a LO mode. If a job of the task τ_i does not complete its execution and its execution time exceeds $C_i(HI)$, the system is regarded as erroneous. The system is initially in a LO mode. The system will switch to a HI mode when the execution time of a task τ_i exceeds $C_i(LO)$.

Correctness criteria. Consider MC sporadic task sets $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ including n independent sporadic tasks. The algorithm scheduling MC sporadic task sets $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ is correct if it should meet the following conditions:

- All tasks end within their deadlines and their execution time does not exceed WCET in a LO mode.
- All HI tasks end within their deadlines and their execution time does not exceed WCET in a HI mode. In addition, all LO tasks (i.e. the criticality level of a task is LO) will be dropped in a HI mode.

B. Power Model

The DVFS mechanism is presented in many of the modern-day processors such as Intel Xscale and AMD A86410. The processor can be operated in variable frequency levels vary the minimum frequency to the maximum frequency $S_{\max}$. The processor frequency is normalized with the maximum frequency. We use the state-of-the-art power model [4] as follows.

$$P = P_{ind} + P_{dynamic}^{\max}(\theta S + S^3) \tag{2}$$

Where P_{ind} is an independent-frequency power which is caused by I/O and memory operation; $P_{dynamic}^{\max}$ is dynamic power which is caused by the circuit activity and it can be normalized to 1; θ is the ratio of static power and dynamic power at the maximum frequency; S is the normalized frequency. The energy consumption E during $[t_1, t_2]$ is equal to $E = \int_{t_1}^{t_2} P dt$. In addition, we assume that the execution time of a task τ_i scales linearly proportional to its normalized frequency S_i [28-29].

TABLE 1
Symbols and notations

Notation	Description
DVFS	Dynamic voltage and frequency scaling
MC	Mixed-criticality
T_i	The minimum inter-arrival separation of the jobs of τ_i
D_i	A relative deadline of τ_i
WCET	Worst case execution time
LO	Low criticality
HI	High criticality
L_i	The criticality level set of τ_i
$C_i(LO)$	The WCET of τ_i in a LO mode
$C_i(HI)$	The WCET of τ_i in a HI mode
$U_{HI}^{LO}(\Gamma)$	The sum of the utilization of HI tasks in Γ in the LO mode
$U_{LO}^{LO}(\Gamma)$	The sum of the utilization of LO tasks in Γ in the LO mode
$U_{HI}^{HI}(\Gamma)$	The sum of the utilization of HI tasks in Γ in the HI mode
τ_{ij}	The jth job of a task τ_i
DFU	A dynamically frequency updating MC algorithm
r_{ij}	The released time of a job τ_{ij}
TS	A subset of Γ, the inter-arrival separation between two successive jobs of a task τ_i in TS is greater than T_i.
U	The current utilization of Γ
S_C	The current frequency of the current executed tasks

IV. Recap of Earliest Deadline First with Virtual Deadlines (EDF-VD)

EDF has poor performance when it is applied to schedule MC sporadic tasks [5-6]. Therefore, the EDF-VD algorithm based on EDF is proposed to MC scheduling. The basic idea of EDF-VD is to shorten the relative deadline of HI tasks which will push HI tasks to finish earlier in a LO mode. The relative deadline of HI task τ_i is set to xT_i in the LO mode. In addition, the relative deadline of HI task τ_i resumes to T_i in a HI mode. The parameter x can be calculated as follows [5]:

$$x \geq \frac{1 - U_{HI}^{LO}(\Gamma)}{U_{LO}^{LO}(\Gamma)} \tag{3}$$

Theorem 1 gives a sufficient condition that EDF-VD scheduling all HI tasks in the HI mode is feasible.

Theorem 1 [5]: The following condition is sufficient for ensuring that EDF-VD successfully schedules all HI tasks in the HI mode:

$$xU_{LO}^{LO}(\Gamma) + U_{HI}^{HI}(\Gamma) \leq 1 \tag{4}$$

An example which consists of two MC sporadic tasks $\tau_1(6,6,LO,2,2)$ and $\tau_2(8,8,HI,1,3)$ is applied to explain EDF-VD. Note that the inter-arrival separation between two successive jobs of τ_i is fixed to T_i. In addition, the jobs of τ_1 and τ_2 are released simultaneously at time 0. According to the method in [7], we compute the parameter x which can be chosen in $[\frac{3}{16}, 1]$. For simplicity, the parameter x is set to 0.5. The system is in a LO mode at the beginning. At time 0, the deadline of τ_{11} and τ_{21} are 6 and 4, respectively. Therefore, τ_{21} begins to execute and finish at time 1. At time 8, τ_{22} begins to execute and it does not end at time 9. The system switches to a HI mode and the jobs LO of τ_1 are dropped after then. Therefore, τ_{22} ends at time 11 and τ_2 will need three time units every period. The detail scheduling can be found in Fig.1.

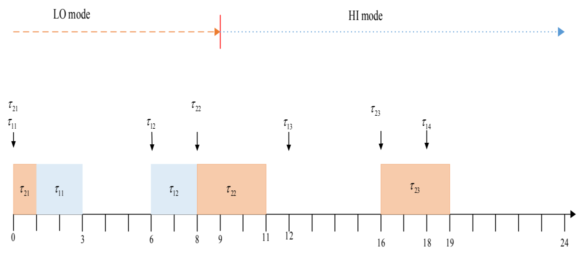

Fig.1. An example of EDF-VD

V. The Proposed Algorithm

In this section, we introduce the motivational example and problem definition in Section V.A. The dynamically updating utilization algorithm and DFU are presented in Section V.B and Section V.C, respectively.

A. Motivational Example and Problem Definition

Some researchers [8-9] focus on power-aware MC sporadic tasks scheduling based on EDF-VD. The inter-arrival separation between two successive jobs of τ_i is fixed to T_i in these studies. In fact, it can be larger than T_i, which leads to calculated frequency is larger than the required frequency. Therefore, there is still much room to further reduce energy consumption, which is exemplified in Table 2 through three MC sporadic tasks.

TABLE 2
The parameter of MC sporadic tasks

Tasks	T_i	D_i	L_i	$C_i(LO)$	$C_i(HI)$
τ_1	6	6	LO	2	2
τ_2	8	8	HI	2	3
τ_3	16	16	LO	4	4

We assume that the job of τ_1 is released at time 0, 10, 20, 32, 42. The job of τ_2 is released at time 0, 12, 20, 34, 42. The job of τ_3 is released at time 0, 24. In addition, we assume that the processor can provide continuous frequencies in [0.3, 1].The example is scheduled in [0, 48] and two power-aware MC sporadic tasks algorithms are applied to explain that there is much room to further reduce energy consumption.

Algorithm A [8] A static optimal solution algorithm—The HI task runs at the frequency S_{HI} in a LO mode and the frequency $S_{\max}$ in a HI mode, and the LO task runs at the frequency S_{LO} in LO modes.

Algorithm B [9] A dynamic solution algorithm—The HI task runs at the frequency S_{HI} in the LO mode and the frequency $S_{\max}$ in the HI mode, and the LO task runs at the frequency S_{LO} in the LO mode. But the frequency S_{HI} and S_{LO} are not fixed. They dynamically change according to the completion of HI tasks.

According to **Algorithm A [8]**, we compute $x = 0.875$, $S_{LO} = 0.86$, and $S_{HI} = 0.90$. The MC sporadic task in Table 2 is scheduled by **Algorithm A** with the scheduling result shown in Fig.2.

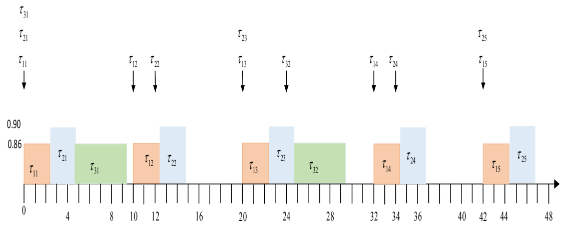

Fig.2. An example of Algorithm A in a LO mode.

Algorithm B [9] first applies the parameter x, S_{LO}, and S_{HI} given in **Algorithm A** to dynamically compute the frequency of both LO tasks and HI tasks. Therefore, the parameter x, S_{LO}, and S_{HI} are equal to 0.875, 0.86, and 0.90, respectively. When τ_{21} finishes at time 2.33, the frequency S_{LO} and S_{HI} are updated as 0.83 and 0.86, respectively. The MC sporadic task in Table 2 is scheduled by **Algorithm B** with the scheduling result shown in Fig.3.

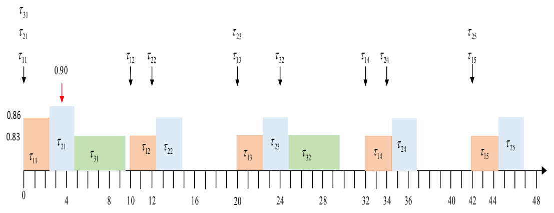

Fig.3. An example of Algorithm B in a LO mode.

As shown in Fig.2, there are some idle intervals such as [9.20, 10], [14.55, 20], [29.20, 32], [36.55, 42], and [46.55, 48]. Moreover, there are also some idle intervals [9.37, 10], [14.74, 20], [29.56, 32], [36.74, 42], and [46.74, 48] in Fig.3. The main reason of such idle intervals is that the inter-arrival separation between two successive jobs a MC sporadic task τ_i is larger than T_i, which leads to lower actual utilization of task τ_i compared with the utilization used in **Algorithm A** [8] and **Algorithm B** [9]. Therefore, we can exploit these idles intervals to further reduce energy consumption.

Problem Definition. Given MC sporadic task sets $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ scheduled by EDF-VD, we should decide the parameter x, S_{LO}, S_{HI} and reclaim the slack time generated from the random arrival of sporadic tasks to dynamically compute S_C to reduce energy consumption in LO mode while meeting **Correctness criteria** of scheduling MC sporadic tasks.

B. The Dynamically Updating Utilization Algorithm

Updating Utilization

```
1. When $\tau_i$ releases a job and $\tau_i \in TS$
2. If $L_i = LO$ then
3.     $U = U + (C_i(LO)/S_{LO})/T_i$
4. Else if the job of $\tau_i$ ends before current time
5.     $U = U + (C_i(LO)/S_{HI})/T_i$
6. Else
7.     $U = U + (C_i(LO)/S_{HI})/T_i + (C_i(HI) - C_i(LO))/T_i$
8. $\tau_i$ is removed from $TS$.
9. When $\tau_i$ does not release a new job at time $r_i + T_i$ and
$\tau_i \notin TS$
      // $r_i$ is the released time of previous job of $\tau_i$
10.  If $L_i = LO$ then
11.      $U = U - (C_i(LO)/S_{LO})/T_i$
12.  Else if the job of a task $\tau_i$ ends before current time
         $U = U - (C_i(LO)/S_{HI})/T_i$
13.  Else
14.      $U = U - (C_i(LO)/S_{HI})/T_i - (C_i(HI) - C_i(LO))/T_i$
15.      $\tau_i$ is added to $TS$.
16. When the processor in an idle mode
17.     $TS$ is set as $\Gamma$ and $U = 0$
18. If $U < 0$ then
19.  $U = 0$
```

We use a similar method given in [11-13] to dynamically update the utilization of MC sporadic task set Γ. It updates according to whether the tasks release jobs. When the task first releases a job, the utilization of MC sporadic task set Γ increases. The dynamically updating utilization algorithm is presented by **Updating Utilization**.

We update the utilization of TS according to **Updating Utilization**. If a new job of τ_i is released and it belongs to TS, we increase the utilization of TS and remove it from TS (line 1-8, **Updating Utilization**). Note that when the HI task τ_i has finished before current time, the increasing utilization of TS will be $(C_i(LO)/S_{LO})/T_i$ (line 4-5, **Updating Utilization**). When τ_i does not release a new job at time $r_i + T_i$ and it does not belong to TS, we decrease the utilization of TS and add it to TS (line 9-15, **Updating Utilization**). When no jobs wait to be scheduled i.e. the processor is in an idle mode, TS is set as Γ and U is set as 0 (line 16-17, **Updating Utilization**).

C. The DFU Algorithm

There are two queues for DFU. Ready queues include currently activated jobs which wait to execute on the processor. Delay queues include the completed jobs or unreleased jobs [10]. At the beginning, all jobs do not be released and they are put into the delay queue. When the job of a task is released, it is moved from delay queues to ready queues. In the initialization step, DFU should compute the parameter x, S_{LO}, and S_{HI} through **Algorithm A** [8].

When the job of a LO task τ_i is released at time t, its deadline is set as $t+T_i$ in both modes. In addition, when the job of a HI task τ_i is released at time t, its deadline is set as $t+xT_i$ in a LO mode and $t+T_i$ at a HI mode, respectively. Released jobs are put into the ready queues to execute. The jobs are scheduled by EDF in the ready queue.

The detail description of DFU is provided in **Scheduling**, **Updating Utilization and Select Frequency**.

Scheduling

```
1. Calculate the parameter $x$, $S_{LO}$, and $S_{HI}$ and set $TS=\Gamma, U=0$.
2. Set ready queue into the empty list and all unreleased jobs are put into the delay queue.
3. If a task releases a job, put it into a ready queue according to EDF.
4. The highest priority task $\tau_j$ is scheduled.
5. Apply Updating Utilization and Select Frequency to dynamically determine the frequency of a task $\tau_j$ ($S_C$).
6. If the execution time of $\tau_j$ exceeds $C_j(LO)/S_C$, the system is changed to a HI mode.
7. If a job of $\tau_j$ finishes its execution.
8.     a job of $\tau_j$ is removed from a ready queue and put into the delay queue.
```

Before scheduling MC sporadic task sets, we first calculate the parameter x, S_{LO}, and S_{HI}. In addition, we set $TS=\Gamma, U=0$ (line 1, **Scheduling**). Moreover, the ready queue and delay are set to the empty list (line 2, **Scheduling**). When τ_i release a job, it will be put into the ready queue according to EDF and the highest priority task τ_j will be first scheduled (line 3-4, **Scheduling**). We should dynamically determine the frequency of τ_j by **Updating Utilization** and **Select Frequency** (line 5, **Scheduling**). When the execution time of τ_j exceeds $C_j(LO)/S_C$, the system will switch to a HI mode (line 6, **Scheduling**). When a job of τ_j finishes its execution, it will be put into the delay queue (line 7-8, **Scheduling**).

We determine the frequency according to **Select Frequency.** The reservation time for HI tasks can be freed in a LO mode when the job of HI tasks finishes (line 1-2, **Select Frequency**). When the system is in a HI mode, the frequency will be set as $S_{\max}$ to ensure that the HI task will end within its deadline (line 5-6, **Select Frequency**). Note that there are no LO tasks in a HI mode. The frequency of LO tasks and HI tasks is determined (line 7-12, **Select Frequency**).

Select Frequency

```
1. If the first job of a HI task $\tau_i$ finishes its execution
2.   $U=U-(C_i(HI)-C_i(LO))/T_i$
3. If $U>1$ then
4.   $U=1$
5. If the system is in a HI mode then
6.     $S_C=S_{\max}$
7. Else if the task is a LO task
8.     $S_C=U*S_{LO}$
9. Else
10.      $S_C=U*S_{HI}$
11. If $S_C<S_{\min}$
12.    $S_C=S_{\min}$
```

The time complexity of **Scheduling**, **Updating Utilization, Select Frequency** is $O(n\log n)$, $O(1)$ and $O(1)$, respectively. Therefore, the time complexity of DFU is $O(n\log n)$.

1) Example of DFU

The MC sporadic task in Table 2 is scheduled by DFU in [0, 48]. We assume that the job of a task τ_1 is released at time 0, 10, 20, 32, 42. The job of a task τ_2 is released at time 0, 12, 20, 34, 42. The job of a task τ_3 is released at time 0, 24. The parameter x, S_{LO}, and S_{HI} are equal to 0.875, 0.86 and 0.90, respectively. In addition, we assume that the processor can provide continuous frequencies in [0.3, 1]. At time 0, jobs of τ_1, τ_2 and τ_3 are released simultaneously. Therefore, τ_{11} executes with the frequency of 0.86 and ends at time 2.33. At the same time, the job τ_{21} executes with the frequency of 0.90 and ends at time 4.55. Due to τ_{21} is the first job of a HI task τ_2, the utilization of Γ decreases (line 1-2, **Select Frequency**). Therefore, τ_{31} begins to execute with the frequency of 0.83. At time 6, the utilization of Γ decreases due to τ_1 does not release a job and it does not belong to TS. Therefore, τ_{31} executes with the frequency of 0.49. In addition, the utilization of Γ decreases at time 8. Thus, τ_{31} executes with the frequency of 0.30. Moreover, the utilization of Γ increases at time 10. Therefore, τ_{31} executes with the frequency of 0.58. In addition, the utilization of Γ increases at time 12. Thus, τ_{31} and τ_{12} execute with the frequency of 0.83 and end at time 12.06 and 14.47, respectively. τ_{22} executes with the frequency of 0.86 at time 14.47. In addition, the utilization of Γ decreases at time 16. τ_{22} executes with the frequency of 0.34 and ends at time 18. At time 20, τ_{13} executes with the frequency of 0.58 and ends at time 23.45. At the same time, τ_{23} executes with the frequency of 0.61. At time 24, τ_3 releases a job and it belongs to TS.

Therefore, τ_{23} executes with the frequency of 0.86 and it ends at time 25.93. τ_{32} executes with the frequency of 0.83 and 0.49 at time 25.93 and 26, respectively. In addition, τ_{32} executes with the frequency of 0.30. At time 32, τ_{14} executes with the frequency of 0.58. In addition, τ_{14} and τ_{32} execute with the frequency of 0.83 and end at time 35.01 and 37.13, respectively. τ_{24} executes with the frequency of 0.86 and ends at time 37.13. The final scheduling result can be found in Fig.4.

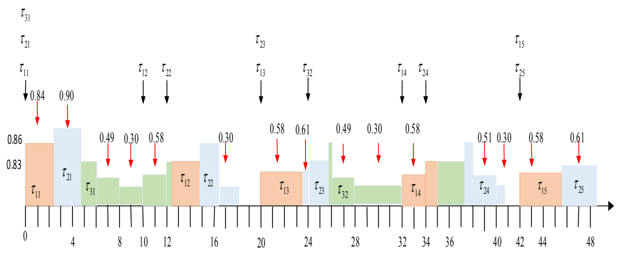

Fig.4. An example of the DFU algorithm a LO mode.

The red arrow and black arrow stand for the frequency and the arriving time of job, respectively. We assume that $P_{ind} = 0.1$. $P_{dynamic}^{\max}$ is normalized to 1 and the θ is set to 0.2 in (2) [4]. In addition, the power is equal to 0.1 when the processor is in an idle status. The energy consumption of **Algorithm A** in Fig.2 and **Algorithm B** in Fig.3 is 32.14 and 30.46, respectively. The energy consumption of DFU is 24.36. Therefore, DFU can save energy up to 24.21% and 20.02% compared with **Algorithm A** and **Algorithm B**, respectively.

2) Schedulability analysis

DFU applies the EDF policy to schedule MC sporadic task sets Γ. If the utilization of Γ is lower or equal to 1, the EDF policy scheduling Γ is feasible. Therefore, we should prove that the utilization of Γ is lower or equal to 1 with dynamically frequency S_C for DFU. The following **Theorem 2** will prove that DFU scheduling Γ is feasible.

Theorem 2. If the utilization of Γ is lower or equal to 1 with the maximum frequency, DFU scheduling Γ with dynamically frequency S_C is feasible.

Proof: Let $\beta_S = \{\beta_{S1}, \beta_{S2}, \cdots, \beta_{Sm}\}$ be the frequency changed intervals where β_{Si} is the intervals between the end of idle intervals and the beginning of the next idle interval. β_S is equal to the hyper-period of Γ if there are no idle intervals (Γ is scheduled in the hyper-period). In addition, we denote $f_S = \{f_{S1}, f_{S2}, \ldots, f_{Sm}\}$ as the set of all frequencies corresponding to the time intervals in β_S. Moreover, we denote $a_U = \{a_{u1}, a_{u2}, \cdots, a_{um}\}$ as the set of the utilization of TS corresponding to the time intervals in β_S. Furthermore, let U_{Si} be the utilization of the interval β_{Si} with the frequency of f_{Si}. U_{Si} can be computed as follows:

$$U_{Si} = \sum_{i=1, \tau_i \in TS}^{n} \frac{C(Si)}{T_i} \tag{5}$$

Where $C(Si)$ is the execution time of τ_i with the frequency of f_{Si} and it can be computed as follows:

$$C(Si) = \begin{cases} C_i(LO)/(a_{ui} \cdot S_{LO}), L_i = LO \\ C_i(LO)/(a_{ui} \cdot S_{HI}), L_i = HI \end{cases} \tag{6}$$

From (5) and (6), we have

$$U_{Si} = \frac{1}{a_{ui}} \left(\sum_{i=1, \tau_i \in TS \wedge L_i = LO}^{n} \frac{C_i(LO)}{T_i \cdot S_{LO}} + \sum_{i=1, \tau_i \in TS \wedge L_i = HI}^{n} \frac{C_i(LO)}{T_i \cdot S_{HI}} \right) \tag{7}$$

The task set TS includes all tasks which do not release jobs at their minimum inter-arrival separation before the beginning of the interval β_{Si}. Therefore, a_{ui} can be computed in (8).

$$a_{ui} = \begin{cases} \sum_{i=1, \tau_i \in TS \wedge L_i = LO}^{n} \frac{C_i(LO)}{T_i \cdot S_{LO}} + \sum_{i=1, \tau_i \in TS \wedge L_i = HI}^{n} \frac{C_i(LO)}{T_i \cdot S_{HI}} \\ (if\ a\ HI\ task\ \tau_i\ has\ finished\ its\ first\ job) \\ \sum_{i=1, \tau_i \in TS \wedge L_i = LO}^{n} \frac{C_i(LO)}{T_i \cdot S_{LO}} + \sum_{i=1, \tau_i \in TS \wedge L_i = HI}^{n} \left(\frac{C_i(LO)}{T_i \cdot S_{HI}} + \frac{C_i(HI) - C_i(LO)}{T_i} \right) \\ (if\ a\ HI\ task\ \tau_i\ doesn't\ finish\ its\ first\ job) \end{cases} \tag{8}$$

The execution time will decrease when the frequency becomes larger. The value of a_{ui} is larger when a HI task τ_i does not finish its first job from (8). The algorithm is feasible if we apply smaller a_{ui} to calculate the frequency. Therefore, the algorithm must be feasible if we use larger a_{ui} to calculate the frequency. Thus, we will use following a_{ui}.

$$a_{ui} = \sum_{i=1, \tau_i \in TS \wedge L_i = LO}^{n} \frac{C_i(LO)}{T_i \cdot S_{LO}} + \sum_{i=1, \tau_i \in TS \wedge L_i = HI}^{n} \frac{C_i(LO)}{T_i \cdot S_{HI}} \tag{9}$$

Applying (9) to (7), we have

$$U_{Si} = 1 \tag{10}$$

The total utilization of time intervals β_S can be computed as a sum of the ratio of products of the utilizations over subintervals times and the interval to the sum of all intervals. It can be calculated as following:

$$U_{\beta_S} = \sum_{j=1}^{m} \frac{\beta_{Si} \cdot U_{Si}}{\sum_{k=1}^{m} \beta_{Sk}} \tag{11}$$

Due to $\sum_{k=1}^{m} \beta_{Sk}$ is a constant, we have

$$U_{\beta_S} = \frac{1}{\sum_{k=1}^{m} \beta_{S_k}} \sum_{j=1}^{m} \beta_{S_i} \cdot U_{Si} \tag{12}$$

Due to $U_{Si} = 1$, we have $U_{\beta_S} = 1$. Therefore, **Theorem 2** is proved.

3) Discussion of the overhead of changing frequency

In this section, we will discuss the overhead of changing frequency. It will generate the time and power overhead when the processor frequency changes. According to [31], the time overhead of changing frequency is equal to $K \cdot |S_i - S_j|$, where K is a constant, S_i and S_j is currently frequency and changed frequency, respectively.

Let O_i be the time overhead of changing frequency for τ_i. O_i is variable and it is determined by the frequency of τ_i. Different jobs of τ_i may have different frequencies during their execution. Therefore, O_i may different for the jobs of τ_i.

The frequency only changes in scheduling point. The so-called scheduling point is the time that the task completes its execution, released jobs, and the job does not be released after the minimum inter-arrival separation. Specially, the frequency is determined by **Updating Utilization.**

We build a queue called α queue to exactly compute O_i. α queue not only records the frequency for the jobs of τ_i in the scheduling point at run-time, but also includes the minimum inter-arrival separation, a relative deadline, WCET in a LO mode and in a HI mode.

When we take the overhead of changing frequency into account, the utilization of task will increase and the slack time will decrease. Therefore, the WECT of τ_i in a LO and a HI mode is equal to $C_i(LO)+O_i$ and $C_i(HI)+O_i$, respectively. When the job of τ_i completes its execution, we then apply $C_i(LO)+O_i$ and $C_i(HI)+O_i$ to instead of WCET in a LO mode and a HI mode for the next job of τ_i, respectively.

We apply the same example in Fig.4 to explain DFU while taking the overhead of changing frequency into account. In addition, we assume that $K=0.1$ [31]. Before scheduling, $O_1=O_2=O_3$. At time 0, τ_{11} executes with the frequency of 0.86 and ends at time 2.33. We compute $O_1=0.014$. At the same time, the job τ_{21} executes with the frequency of 0.90 and ends at time 4.55. We compute $O_2=0.004$. Therefore, we add this overhead to re-compute frequency. τ_{31} begins to execute with the frequency of 0.83. At time 6, it executes with the frequency of 0.49. In addition, it executes with the frequency of 0.30 and 0.58 at time 8 and 10, respectively. Moreover, it executes with the frequency of 0.83 at time 12 and ends at time 12.06. We compute $O_3=0.113$. τ_{12} executes with the frequency of 0.83 and ends 14.47. Then, we compute $O_1=0$. τ_{22} executes with the frequency of 0.86 at time 14.47. It executes with the frequency of 0.34 at time 16 and ends at time 18. We compute $O_2=0.055$. At time 20, τ_{13} executes with the frequency of 0.58 and ends at time 23.45. We compute $O_1=0.024$. At the same time, τ_{23} executes with the frequency of 0.61. It executes with the frequency of 0.88 at time 24 and ends at time 25.89. We compute $O_2=0.03$. τ_{32} executes with the frequency of 0.84 at time 25.89. In addition, it executes with the frequency of 0.50 and 0.30 at time 26 and 28, respectively. At time 32, τ_{14} executes with the frequency of 0.59. In addition, it executes with the frequency of 0.84 and ends at time 34.98. We compute $O_1=0.054$. τ_{32} executes with the frequency of 0.84 at time 34.98 and ends at time 37.02. We compute $O_3=0.058$. τ_{24} executes with the frequency of 0.88 and 0.53 at time 37.02 and 38, respectively. At time 40, it executes with the frequency of 0.30 and ends at time 40.27. We compute $O_2=0.062$. τ_{15} executes with the frequency of 0.58 at time 42 and ends at time 45.45. We compute $O_1=0.028$. τ_{25} executes with the frequency of 0.61 at time 45.45 and ends at time 48.73. We compute $O_2=0.003$.

The energy overhead of changing frequency is 0.05. The energy consumption of DFU while considering the overhead of changing frequency is 24.65. Therefore, the total energy consumption of DFU is 24.70. DFU can save energy up to 23.15% and 18.91% compared with **Algorithm A** and **Algorithm B**, respectively.

4) A real-world application

In this section, we discuss the application of DFU in a real-world. Flight Management System (FMS) is an MC system in a real-world. FMS includes the localization and flight plan tasks (DO-178B level B and level C, where B corresponds to the HI criticality and C corresponds to the LO criticality) [32]. In fact, FMS includes seven HI tasks and four LO tasks. In addition, the tasks of FMS are also sporadic tasks with implicit-deadlines, which is consistent with the task model in this paper. Moreover, the processor for FMS only provides discrete frequencies. However, DFU assumes that the processor can provide continuous frequencies. If the processor for FMS does not provide frequency given by DFU, we can apply the next higher frequency or two adjacent frequencies to solve this problem. Therefore, DFU can easily apply to FMS.

VI. Simulation Experiment

We apply extensive simulation experiment to evaluate the effectiveness of our proposed approach. The simulation experiments are performed on a MC sporadic task scheduling simulator written by C language and based on the EDF policy. Three algorithms are implemented in the simulator.

Algorithm A [8]. A static optimal solution algorithm.

Algorithm B [9]. A dynamic solution algorithm.

DFU. It can dynamically compute S_{LO} and S_{HI}. In addition, it can exploit the time reservation for HI tasks in the LO mode.

Based on the well-known MC task generation scheme in [3, 8, 14]. The uniform distribution method is applied to generate a synthetic random task set. The synthetic random task set includes two LO tasks and two HI tasks [8]. The minimum inter-arrival separation of the jobs of τ_i (i.e. T_i) can be randomly chosen within the range of $[10,100]$. $C_i(LO)$ is randomly chosen within the range of $[1,T_i]$. $C_i(HI)$ is equal to $C_i(LO)$ for all LO tasks. $C_i(HI)$ is randomly chosen within the range of $[C_i(LO),T_i]$ for all HI tasks. The value of $U_{LO}^{LO}(\Gamma)$, $U_{HI}^{LO}(\Gamma)$, and $U_{HI}^{HI}(\Gamma)$ does not exceed the given value by modifying $C_i(LO)$ and $C_i(HI)$. In addition, we generate 100 synthetic random task sets and measure average energy consumption. Moreover, we set the simulation time to 10^6 time units because different task sets have different hyper-periods. We assume that $P_{ind}=0.1$. $P_{dynamic}^{\max}$ is normalized to 1 and the θ is set to 0.2 in (2) [4]. In addition, we will analyze different static power situations affect our algorithm by modifying the value of θ. Note that we focus on the energy consumption of the algorithm in the LO mode. Moreover, the energy consumption of **Algorithm A** is applied as baseline i.e. the energy consumption of other algorithm is normalized with **Algorithm A**.

A. Effect of $U_{LO}^{LO}(\Gamma)$

$U_{HI}^{HI}(\Gamma)$ is fixed to 0.5 and the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ is set to 1.3. We vary $U_{LO}^{LO}(\Gamma)$ from 0.05 to 0.45, stepped by 0.5 and investigate $U_{LO}^{LO}(\Gamma)$ affecting on the energy consumption of the algorithm. The experimental result is shown in Fig.5.

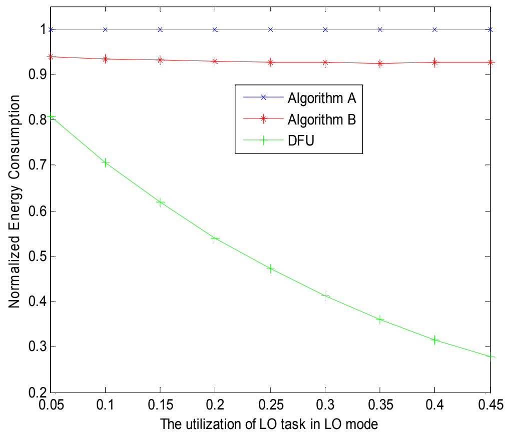

Fig.5. The effect of $U_{LO}^{LO}(\Gamma)$

The normalized energy consumption of **Algorithm B** is not sensitive to $U_{LO}^{LO}(\Gamma)$ and the normalized energy consumption of DFU is highly dependent on $U_{LO}^{LO}(\Gamma)$ in Fig.5. The energy consumption of all algorithms is positively related to $U_{LO}^{LO}(\Gamma)$. As $U_{LO}^{LO}(\Gamma)$ increases, the energy consumption of **Algorithm A** and **Algorithm B** grows with similar magnitude, whereas DFU grows much slower. In addition, we normalize energy consumption with respect to the energy consumption of **Algorithm A**. Furthermore, the normalized energy consumption of DFU and **Algorithm B** is less than that of **Algorithm A**. DFU consumes 46.20% less energy than that of **Algorithm B**.

B. Effect of $U_{HI}^{HI}(\Gamma)$

$U_{LO}^{LO}(\Gamma)$ is fixed to 0.3 and the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ is set to 1.3. We vary $U_{HI}^{HI}(\Gamma)$ from 0.05 to 0.6, stepped by 0.5 and investigate $U_{HI}^{HI}(\Gamma)$ affecting on the energy consumption of the algorithm. The experimental result is shown in Fig. 6.

The normalized energy consumption of **Algorithm B** and DFU is sensitive to $U_{HI}^{HI}(\Gamma)$ in Fig.6. As $U_{HI}^{HI}(\Gamma)$ increases, the energy consumption of **Algorithm B** and DFU decreases. This is because the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ is fixed and the slack time will increase when $U_{HI}^{HI}(\Gamma)$ increases. **Algorithm B** and DFU can exploit this slack time to reduce processor frequency. In addition, the normalize energy consumption of DFU is lower than that of **Algorithm B and Algorithm A.** The reason is that DFU can dynamically update the utilization of Γ to dynamically reduce processor frequency. DFU consumes 34.29% less energy than that of **Algorithm B**.

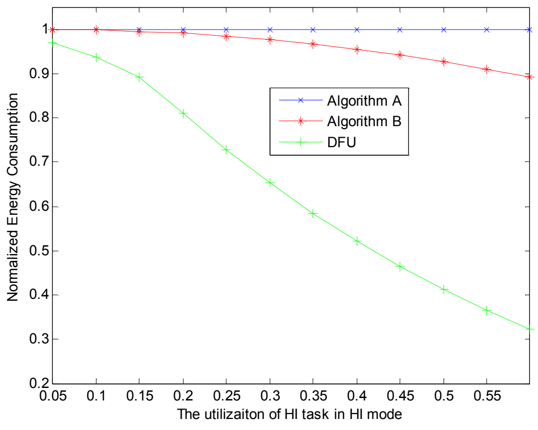

Fig.6. The effect of $U_{HI}^{HI}(\Gamma)$

C. Effect of the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$

$U_{LO}^{LO}(\Gamma)$ and $U_{HI}^{HI}(\Gamma)$ are fixed to 0.4 and 0.5, respectively. We vary the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ from 1.1 to 1.9, stepped by 0.1 and investigate the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ affecting on the energy consumption of the algorithm. The experimental result is shown in Fig. 7.

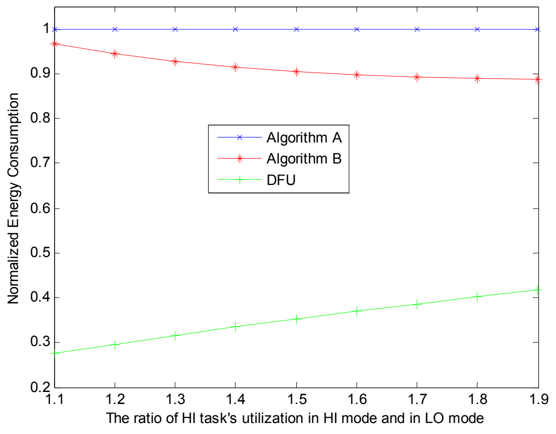

Fig.7. The effect of the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$

The normalized energy consumption of **Algorithm B** and DFU is sensitive to the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ in Fig.7. As the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ increases, the normalized energy consumption of **Algorithm B** decreases. This is because $U_{HI}^{HI}(\Gamma)$ is fixed and $U_{HI}^{LO}(\Gamma)$ decreases when the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ increases. Therefore, the execution time of tasks will decrease in the LO mode and **Algorithm B** can exploit slack time generated from HI tasks to dynamically reduce processor frequency. In addition, the normalized energy of **Algorithm B** increases when the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ increases. The energy consumption of DFU decreases much slower than that of **Algorithm A**. In addition, we normalize energy consumption with respect to the energy consumption of **Algorithm A**. Moreover, the dynamically utilization of Γ changes slowly, which leads to slowly changing of processor frequency. In short, DFU consumes 61.51% less energy than that of **Algorithm B**.

D. Effect of static power

$U_{LO}^{LO}(\Gamma)$ and $U_{HI}^{HI}(\Gamma)$ are fixed to 0.3 and 0.6, respectively. In addition, the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ is fixed to 1.3. We vary the ratio of static power and dynamic power from 0.2 to 0.55, stepped by 0.05 and investigate the ratio of static power and dynamic power affecting on the energy consumption of the algorithm. Moreover, the energy consumption is normalized with respect to **Algorithm A** when the ratio of static power and dynamic power is equal to 0.55. The experimental result is shown in Fig.8.

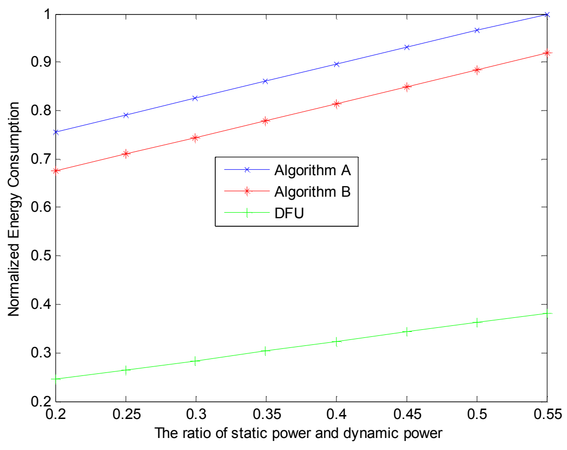

Fig. 8. The effect of static power

As shown in Fig.8, the normalized energy consumption of algorithms increases when the ratio of static power and dynamic power increases. The reason is that the execution time of tasks and processor frequency do not change. The energy consumption is determined by the execution time of tasks, processor frequency, and power. The power will increase when the ratio of static power and dynamic power increases. Therefore, the energy consumption will increase. In this case, DFU consumes 60.87% less energy than that of **Algorithm B**.

E. Consider the overhead of changing frequency

$U_{HI}^{HI}(\Gamma)$ is fixed to 0.5 and the ratio of $U_{HI}^{HI}(\Gamma)$ and $U_{HI}^{LO}(\Gamma)$ is set to 1.3. We vary $U_{LO}^{LO}(\Gamma)$ from 0.05 to 0.45, stepped by 0.5. In addition, the energy consumption of DFU is used as baseline. The energy consumption of DFU while considering the overhead of changing frequency is represented as DFUO in Fig.9. The experimental result is shown in Fig. 9.

As shown in Fig.9, the normalized energy consumption of DFUO is higher than that of DFU. This is because DFUO considers the overhead of changing frequency. In addition, the DFUO has little added energy consumption compared with DFU. It means that the overhead of changing frequency has a small impact on the performance of DFU. In addition, the overhead of changing frequency is determined by the frequency of DFU. The frequency of DFU is determined by the utilization of tasks. When $U_{LO}^{LO}(\Gamma)$ varies, the overhead of changing frequency also changes. All in all, DFUO has good energy saving performance compared with previous studies.

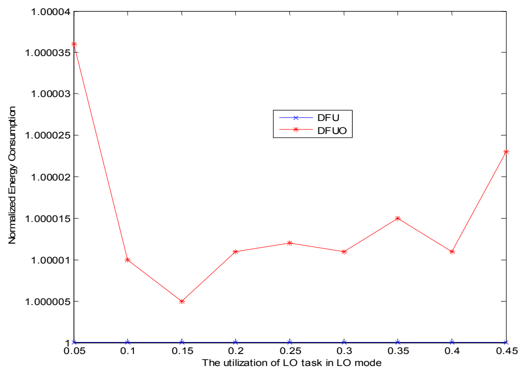

Fig.9. Consider the overhead of changing frequency

VII. Conclusion

We focus on scheduling energy aware MC sporadic tasks. Firstly, we propose DFU. It based on EDF scheme can exploit the slack time generated from the HI task in the LO mode to reduce processor frequency. In addition, it also can dynamically update the utilization of Γ to further reduce processor frequency. Secondly, we analyze the scheduling feasibility of DFU. Finally, the simulation experiments are conducted to evaluate the performance of DFU. The experimental results show that DFU has better energy saving effects than other algorithms.

DFU assumes that the MC sporadic tasks are independent. We will focus on the dependent MC sporadic tasks in the future work.

Acknowledgment

This work has been supported by the Natural Science Foundation of Fujian Province of china under Grant 2019J01080.

References

[1]S Baruah, A Burns, R Davis. Response-Time Analysis for Mixed Criticality Systems. Real-Time Systems Symposium, 2012, pp. 34-43.
[2]S Baruah, B Chattopadhyay. Response-time analysis of mixed criticality systems with pessimistic frequency specification. In the 19th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2013, pp. 237-246.
[3]Q Zhao et al. Schedulability analysis and stack size minimization with preemption thresholds and mixed-criticality scheduling. Journal of Systems Architecture, vol 83, 57-74, 2018.
[4]A Taherin, M Salehi, A Ejlali. Reliability-Aware Energy Management in Mixed-Criticality Systems. IEEE Transactions on Sustainable Computing, vol 3, no 3, 195-208, 2018.
[5]S Baruah et al. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In 2012 24th Euromicro Conference on Real-Time System, 2012, pp. 145-154.
[6]D Müller,A Masrur. The schedulability region of two-level mixed-criticality systems based on EDF-VD. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014, pp. 1-6.
[7]S Baruah, Z Guo. Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor. Real-Time Systems Symposium, 2014, pp. 31-40.
[8]P Huang et al. Energy efficient dvfs scheduling for mixed-criticality

systems. Proceedings of the 14th International Conference on Embedded Software, 2014, pp. 11-20.
[9]I Ali, J Seo, K Kim. A dynamic power-aware scheduling of mixed-criticalit real-time systems. IEEE International Conference on Computer and Information Technology, Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015, pp. 438-445.
[10]Y Zhang, R Guo. Power-aware fixed priority scheduling for sporadic tasks in hard real-time systems. Journal of Systems and Software, vol 90, no 2, 128-137, 2014.
[11]A Qadi., S Goddard., S Farritor. A dynamic voltage scaling algorithm for sporadic tasks. In: Proceedings of the 24th Real-Time System Symposium, 2003, pp. 52-62.
[12]Y Zhang, C Wang, C Lin. Energy-aware sporadic tasks scheduling with shared resources in hard real-time systems. Sustainable Computing Informatics & Systems, vol 15, 52-62, 2017.
[13]Y Zhang, C Xu. Low power fixed priority scheduling sporadic task with shared resources in hard real time system. Microprocessors and Microsystem, vol 45, 164-175, 2016.
[14]Z Li et al. Reliability guaranteed energy minimization on mixed-criticality systems. Journal of Systems and Software, vol 112, 1-10, 2016.
[15]S Baruah et al. The Preemptive Uniprocessor Scheduling of Mixed-Criticality Implicit-Deadline Sporadic Task Systems. 24th Euromicro Conference on Real-Time Systems (ECRTS), 2012, pp.145-154.
[16]S Baruah, Z Guo. Mixed-Criticality Scheduling upon Varying-Speed Processors. IEEE34th Real-Time System Symposium (RTSS 2013), 2013, pp. 68-77.
[17]S Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In the 28th IEEE Real-Time Systems Symposium (RTSS), 2007, pp. 239-243.
[18]Y Chen, G Kang, H Xiong. Generalizing fixed-priority scheduling for better schedulability in mixed-criticality systems. Information Processing Letters, vol 116, no 8, 508-512, 2016.
[19]Q Zhao et al. Schedulability Analysis and Stack Size Minimization with Preemption Thresholds and Mixed-Criticality Scheduling. Journal of Systems Architecture, vol 83, 57-74, 2018.
[20]Y Chen et al. Efficient schedulability analysis for mixed-criticality systems under deadline-based scheduling. Chinese Journal of Aeronautics, vol 27, no 4, 856-866, 2014.
[21]G Chen et al. Utilization-Based Scheduling of Flexible Mixed-Criticality Real-Time Tasks. IEEE Transactions on Computers, vol 67, no 4,543-558, 2018.
[22]M HorHorng, C Huang, Y Kuo, Hu. Scheduling sporadic, hard real-time tasks with resources. Proceedings of 3rd International Conference on Innovative Computing Information and Control, 2008, pp. 84-87.
[23]Y Zhang, C Wang, J Liu. Energy aware fixed priority scheduling for real time sporadic task with task synchronization. Journal of Systems Architecture, vol 83, 12-22, 2018.
[24]Y Zhang. System level fixed priority energy management algorithm for embedded real time application, Microprocessors and Microsystems, vol 64, 170-177, 2019.
[25]Y Guo et al. Exploiting primary/backup mechanism for energy efficiency in dependable real-time systems. Journal of Systems Architecture, vol 78, 68-80, 2017.
[26]M Fakih et al. SAFEPOWER project: Architecture for Safe and Power-Efficient Mixed-Criticality Systems. Microprocessors and Microsystems, vol 52, 89-105, 2017
[27]A Easwaran. 2013. Demand-based scheduling of mixed-criticality sporadic tasks on one processor. In Proceedings of the 34th Real-Time Systems Symposium (RTSS), 2013, pp.78-87.
[28]Y Zhang, H Li. Energy aware mixed tasks scheduling in real-time systems, Sustainable Computing-Informatics & Systems, vol 23, 38-48, 2019.
[29]Y Zhang. Energy-aware mixed partitioning scheduling in standby-sparing systems, Computer Standards & Interfaces, vol 61, 129-136, 2019.
[30]B, Ashikahmed et al. Precise scheduling of mixed-criticality tasks by varying processor speed. "In Proceedings of the 27th International Conference on Real-Time Networks and Systems, 2019, pp.123-132.
[31]Y Zhang, R Guo. Low power scheduling algorithms for sporadic task with shared resources in hard real-time systems, The Computer Journal, vol 58, no 7, 1585-1597, 2015.
[32]P Huang, G Giannopoulou, N Stoimenov, L Thiele. Service adaptions for mixed-criticality systems. In ASP-DAC, 2014, pp. 125-130.

Yi-wen Zhang received his B.E degree in Department of Mathematics, Yangtze Normal University, Chongqing, China in 2010 and PH.D from University of Chinese Academy of Sciences in 2016. Now he is associate professor in College of Computer Science and Technology, Huaqiao University, Xiamen, China. His current research interests include real-time system and low-power design.