2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

A Dynamic Power-aware Scheduling of Mixed-Criticality Real-Time Systems

Ijaz Ali, Jun-ho Seo, and Kyong Hoon Kim
Department of Informatics
Gyeongsang National University
501 Jinju-daero, Jinju 660-701, South Korea
Email: ijazali1984@yahoo.com, {joy2net, khkim}@gnu.ac.kr

***Abstract*—Recent safety-critical real-time embedded systems have enabled multiple real-time applications with different criticalities to run in a single system. Thus, many studies have been done not only on providing timing guarantee for tasks having different criticality levels, but also on other required functional properties such as energy reduction. Since many processors are equipped with dynamic voltage and frequency scaling (DVFS) capacity, energy can be saved by reducing the processor frequency. In this paper, we provide a new dynamic power-aware mixed-criticality real-time scheduling. The proposed scheme can reduce energy more by using slack time for reserved time for high-criticality tasks. Throughout simulation results, we show that the proposed scheme reduces more energy and that the optimal frequency levels in the static scheme are not optimal in the dynamic scheme.**

I. Introduction

As recent computing technology has been developed rapidly, safety-critical real-time embedded systems provides multiple functionalities in a single system. Such a system having applications of different functionalities or criticalities is called mixed-criticality system. This integration of functionalities is considered by many designers due to power, space and weight issues. For example, multiple functionalities of different criticalities are integrated on the same platform in ARINIC-653 for modern aircraft systems [1]. Mixed-criticality systems require significant challenges to the design of safety-critical real time embedded systems.

A mixed-criticality real-time system is generally defined as a set of multiple components with different criticalities [2]. Most of recent work on mixed-criticality real-time systems has focused on scheduling issues under two criticalities: *low* and *high* mode. Thus, a task has either low-criticality or high-criticality. The system starts from low mode where all tasks execute low-mode execution times. When any high-criticality task does not finish its execution time, the system changes its mode to high mode in which only high-criticality tasks are executed. Many recent work has been done on scheduling of mixed-criticality real-time tasks [3], [4], [5], [6], [7].

As power-consumption becomes an important issue in recent real-time embedded systems, lots of work has been suggested to reduce energy with the real-time constraints. One of common techniques for reducing dynamic power consumption is based on Dynamic Voltage Frequency Scaling (DVFS) scheme which scales the processor frequency dynamically. Since there are tradeoffs between execution time and power consumption, many studies have been done on RT-DVFS (Real-Time DVFS) problem [8] for aperiodic and periodic tasks. RT-DVFS schemes utilize slack times to adjust the processor speed without violating tasks deadlines.

Thus, in this paper, we provide an efficient power-aware scheduling scheme for mixed-criticality real-time systems. Only a few studies have been done on power-aware mixed-criticality real-time systems. In [9], a processor speed degradation scheme is provided for a given mixed-criticality aperiodic real-time jobs. In [10], they defined an optimization problem of power-aware scheduling in mixed-criticality real-time systems under continuous frequency level. During the run-time, each task is executed on the derived frequency level. Thus, we improve the previous static scheme in a dynamic way where the frequency level is adjusted below the derived frequency level for the purpose of power reduction. The main contribution of this work is (i) providing a dynamic power-aware scheduling scheme of mixed-criticality real-time tasks and (ii) analysis of the provided scheme throughout simulation results.

The remainder of this paper is organized as follows. Section II defines related work and problem description. In Section III, the system model and research motivation are provided. In Section IV, we explain the proposed power-aware scheduling scheme. We evaluate the proposed scheme using simulation results in Section V, and conclude the paper in Section VI.

II. Related Work and Problem Description

Mixed-criticality scheduling is introduced by Vestal [3]. For scheduling tasks in mixed-criticality systems, he extended fixed-priority response-time analysis of sporadic tasks for this purpose [3]. Later on, Dorin et al. [11] proved in their research work that Vestal algorithm is optimal only for fixed-task-priority preemptive algorithms. Baruah et al. [4] also introduced fixed-priority schemes, based on response-time analysis for increasing the schedulability of tasks in the mixed-criticality systems. An efficient heuristic scheduling strategy, called Own Criticality Based Priority (OCBP) is proposed by Baruah et al. [12]. This strategy uses a variant of the Audsley approach for priorities assignment to the mixed-criticality jobs, based on their own criticality levels. In Audsley approach [13], priorities are assigned to the jobs in a recursive manner from lowest to highest scheduling priority, so that priority assigned to first task is the lowest priority task and so on.

The scheduling complexity of mixed-criticality system is investigated by Baruah, et al. These authors proved in the research work that even if all mixed-criticality jobs are released

978-1-5090-0154-5/15 $31.00 © 2015 IEEE
DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.63

at the same time, the optimal schedulability of such jobs set is strongly NP-hard, [5]. Thus, it is required to investigate optimal scheduling strategies for mixed-criticality systems that perform well in practice. The heuristic OCBP strategy is extended to mixed-criticality sporadic task systems by Li and Baruah (LB) [6]. But it is computationally expensive, because when the job releases deviate from the assumed as-early-as possible job release (off-line) pattern during execution then each time new priorities are computed and assigned again to the jobs in a busy period (run-time). Later on, Guan et al. [7] presented a scheduling strategy known as PLRS (Priority List Reuse Scheduling) which is also based on OCBP scheduler.

Power-aware scheduling techniques are generally classified into *static* and *dynamic* techniques. Furthermore, the techniques can also be divided based on whether or not they consider *discrete* or *continuous* frequency scaling. Static techniques determine the optimal processor frequency before the system runs. These schemes are also referred to as *offline* schemes. Aydin et al. [14] provided an optimal static DVFS scheme based on the Earliest Deadline First (EDF) scheduling policy for periodic real-time tasks with potentially different power consumption characteristics. Gruian [15] proposed a stochastic voltage scheduling scheme that take advantage of the probability distribution of the execution pattern of each task. Jejurikar and Gupta [16] determined sub-optimal slowdown factors for a set of periodic real-time tasks. Instead of focusing only the processor, Zhang and Xu [17] considered a system-wide energy consumption model for both periodic and sporadic tasks running on processors with discrete speed levels. A similar energy model is also assumed by Yun et al. [18] who presented an optimal static scheme for the energy reduction of multiple components with individually adjustable frequencies such as the processor, the system bus and the main memory. The work in this paper falls in this categories, while assuming only real-world processors with discrete frequency levels.

On the other hand, dynamic techniques, also referred to as *online* techniques, determine the processor speed levels at runtime by exploiting useful information like variations in the actual workload or unused processor slack times. Aydin et al. [19] presented a reclaiming and a speculative DVFS algorithm for periodic tasks that complete before their worst-case execution time. Considering the discrete frequency scaling, Mejia-Alvarez et al. [20] proposed an adaptive server to determine the processor speed level at runtime. Marinoni and Buttazzo [21] suggested to combine elastic scheduling with DVFS to fully utilize the resources on processors with discrete frequencies. Wang et al. [22] proposed a dynamic slack time reclamation approach that considers both the leakage power consumption and the voltage scaling overhead. Mei et al. [23] applied the cycle conserving approach [24] to reduce the power consumption of a real-time system composed of sporadic tasks.

Only a few recent studies have been done on power-aware mixed-criticality real-time systems. In [9], they derived possible processor speed degradation for a given mixed-criticality jobs. They first introduced the problem of power-aware scheduling of mixed-criticality real-time systems and considered only a set of jobs, not with periodic tasks. In [10], they formulated an optimization problem of power-aware mixed-criticality scheduling for a set of periodic tasks under continuous frequency level. The algorithm is based on EDF-VD [25] and provides an optimal virtual deadlines and frequency levels of tasks. They do not adjust the derived frequency levels of tasks during the run-time. In this paper, we assume a discrete frequency-level processor and dynamically adjust the frequency level of tasks in order to reduce more power consumption.

III. System Model and Research Motivation

A. Task Model

The task model in this paper is based on Mixed-Criticality task model [3] so that a task τ_i consists of four parameters $(P_i, C_i(LO), C_i(HI), X_i)$ as follows.

- P_i: the task period
- $C_i(LO)$: the worst-case execution time in low level criticality
- $C_i(HI)$: the worst-case execution time in high level criticality
- X_i: the criticality (LO or HI)

A task τ_i is a periodic real-time task, which implies that it releases a job every P_i time units. We denote $\tau_{i,j}$ as the j-th job or instance of task τ_i. In the mixed-criticality real-time systems, a task has either low-criticality or high-criticality so that X_i denotes the task criticality (LO for low-criticality and HI for high-criticality). In low-criticality mode, each task requires $C_i(LO)$ time units every its period. However, after the mode switch from low to high occurs, all low-criticality tasks are dropped in the system and each high-critical task τ_i requires $C_i(HI)$ execution times every period ($C_i(HI) \leq C_i(LO)$).

In mixed-criticality real-time systems, the mode switch is an important event since it brings the system mode into high-criticality mode from low mode. High-criticality implies that the system should execute high-criticality tasks rather than low-criticality ones. Thus, in high-criticality mode, all low-criticality tasks are removed from the scheduling queue and only high-criticality tasks are executed. The mode switch event occurs when any high-criticality task does not finish its $C_i(LO)$ execution times.

Let us consider two mixed-criticality tasks of $\tau_1(6, 3, 3, LO)$ and $\tau_2(8, 1, 4, HI)$. As shown in Figure 1, two tasks execute their low-mode execution times in low-criticality mode at the beginning. That is, τ_2 requires only one time unit in low-mode as in the first period in Figure 1. Let us assume that τ_2 does not finish its low mode job at time 10.

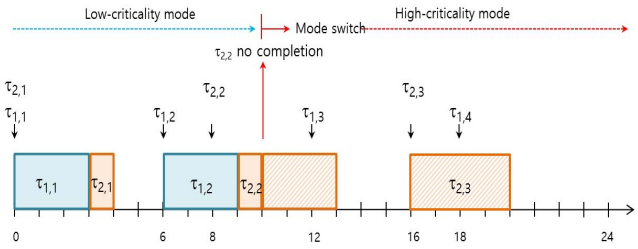


Fig. 1. An example of mixed-criticality real-time scheduling

This event induces a mode switch event to the system so that the low-criticality task τ_1 is not scheduled after then. In addition, the high-criticality task τ_2 requires four time units every period.

B. Power Model

In this paper, we consider mixed-criticality real-time tasks running on a single processor system. We assume the processor with Dynamic Voltage and Frequency Scaling (DVFS) mechanism allowing it to dynamically adjust its operating frequency. The processor is assumed to have a finite set F of m discrete speed levels, where m represents the total number of frequencies supported by the processor.

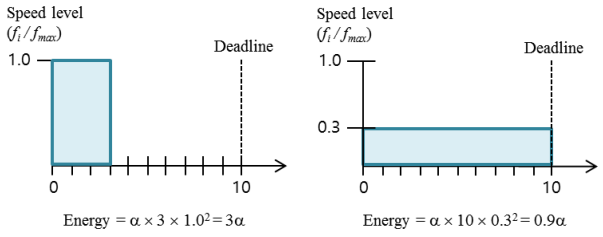
Let us consider an application with execution time t running on a processor with maximum frequency $f_{\max}$. If the processor is scaled to run at a frequency level f ($0 < f \leq f_{\max}$), the relative processor speed level s is defined by $f/f_{\max}$. Thus, the application execution time on the processor with a speed level s is defined by t/s, where t is the execution time with the maximum processor speed level $s_{\max}$. That is, the application execution time is slowed down by a factor of s.

The power consumption of a processor core is composed of dynamic and static power. In this work, we only consider the dynamic power consumption, as it is a dominating factor in the total power consumption [26] and is managed by the DVFS scheme. The dynamic power is generally in proportion to the cubic of the frequency f. If an application execution time is given by t under the maximum processor speed $s_{\max}$, then its energy consumption E assuming the processor speed level s is defined by

$$E = \alpha \cdot \frac{t}{s} \cdot s^3 = \alpha \cdot t \cdot s^2, \tag{1}$$

where α is a coefficient. In this paper, we assume $\alpha = 1$ for the sake of simplicity.

The basic idea of DVFS scheme in real-time systems is shown in Figure 2. Let us assume that a real-time application requires three execution times with the deadline 10 unit times, as shown in Figure 2(a). As long as the application does not miss its deadline, we can slow down the frequency level as shown in Figure 2(b). Since the power consumption depends on the square of speed level, the DVFS scheme is useful for power-aware real-time task scheduling.



(a) Maximum frequency (b) Appropriate frequency

Fig. 2. DVFS for real-time tasks

C. Recap of Power-aware EDF-VD

In this subsection, we describe a brief explanation of the previous work on power-aware mixed-criticality tasks scheduling [10]. The base scheduling algorithm is EDF-VD [6] which assigns the virtual deadline of each high-criticality task in low mode and schedules all tasks based on EDF (Earliest Deadline Frist) until the mode switch event occurs. In EDF-VD, the value of x in a system determines the virtual deadline VD_i from $P_i \times x$.

In [10], they defined a new problem of power-aware mixed-criticality tasks scheduling based on EDF-VD under the assumption of continuous frequency level. The problem is to minimize an energy function which satisfies the following two equations:

$$\begin{gathered} \frac{U_{HI}^{LO}}{x} + U_{LO}^{LO} \leq 1 \\ \sum_{\tau_i \in T_{HI}} \frac{C_i(LO)/f^{HI}}{P_i} \cdot \frac{1}{x} + \sum_{\tau_i \in T_{LO}} \frac{C_i(LO)/f^{LO}}{P_i} \leq 1 \end{gathered} \tag{2}$$

$$\begin{gathered} U_{HI}^{HI} + xU_{LO}^{LO} \leq 1 \\ \sum_{\tau_i \in T_{HI}} \frac{C_i(HI)}{P_i} + x \cdot \sum_{\tau_i \in T_{LO}} \frac{C_i(LO)/f^{LO}}{P_i} \leq 1 \end{gathered} \tag{3}$$

where T_{HI} and T_{LO} are two disjoint sets of high-criticality tasks and low-criticality tasks, respectively, in a given mixed-criticality real-time task set. In the above equations, f^{HI} and f^{LO} denote the optimal frequency levels of high-criticality tasks and low-criticality tasks in low mode. Although they assumed continuous frequency level and different energy function, they formulated a convex problem and provided an optimal solution to derive x, f^{LO}, and f^{HI}.

Let us consider a task set in Table I [10] as an example. Figure 3 shows the unit power consumption for various x using the energy function of Equation (1). We assume that the frequency levels are discrete from 0.4 to 1.0 with the interval of 0.1. As shown in Figure 3, the optimal x is 0.625 where f^{HI} and f^{LO} are 0.7 and 0.5 in each. Figure 4 shows the scheduling results of three tasks until the LCM (Least Common Multiple) of periods by EDF-VD with f^{HI} and f^{LO}.

D. Research Motivations

In this paper, we consider power-aware scheduling under discrete frequency levels. As discussed in the previous subsection, we can use the previous approach in [10] to derive the optimal frequency levels of mixed-criticality tasks. However, as shown in Figure 4, there are still much room for further reduction.

TABLE I. AN EXAMPLE OF MIXED-CRITICALITY REAL-TIME TASKS

	P_i	$C_i(LO)$	$C_i(HI)$	X_i
τ_1	8	2	5	HI
τ_2	12	1	1	LO
τ_3	16	2	2	LO

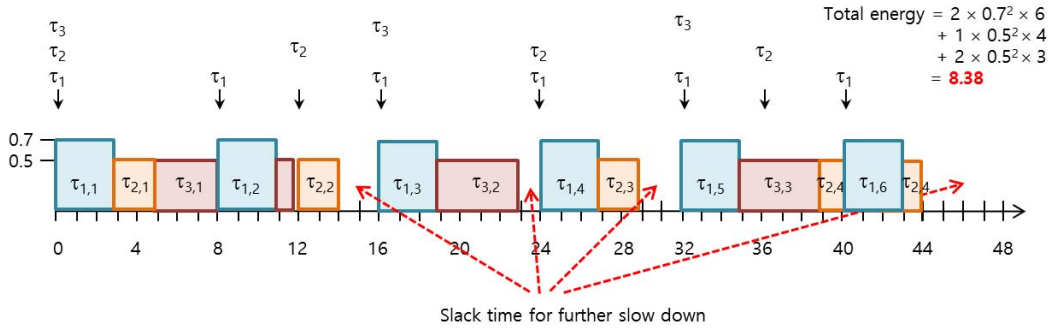


Fig. 4. An example of DVFS scheduling of mixed-criticality real-time tasks by previous scheme

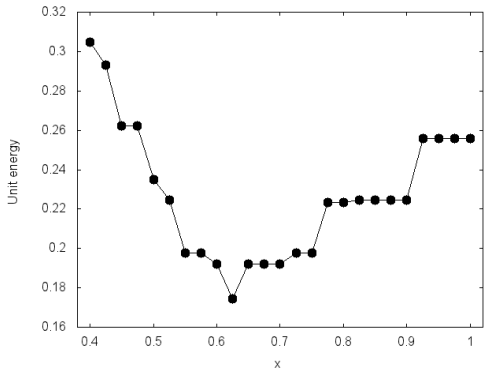


Fig. 3. The unit energy for various x

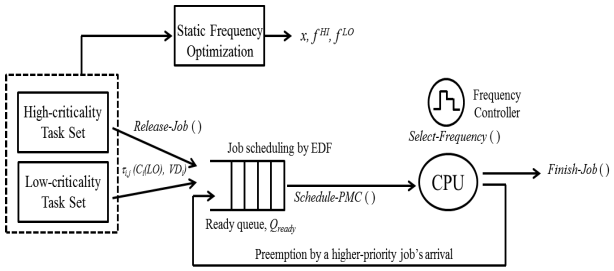
The main reason of such slack time is due to time reservation for high-criticality tasks. In EDF-VD, the virtual deadline VD_i of high-criticality task τ_i is less than or equal to $P_i - (C_i(HI) - C_i(LO))$ since the system reserves the minimum scheduling time in case of mode switch. This reservation time is unused if high-criticality tasks finish their low-mode execution time without any delay. Thus, in this paper, we enhance the previous static scheme by adjusting the frequency level of each task dynamically.

IV. The Proposed Scheme

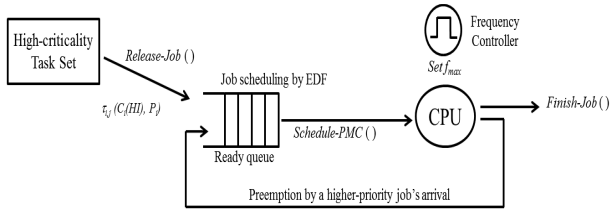
A. Dynamic Power-aware DVFS Scheme for MC Tasks

The proposed power-aware DVFS scheduling system adjusts the CPU frequency level dynamically based on the statically derived frequency levels of tasks. In the initialization step, we derive x, f^{HI}, and f^{LO} of mixed-criticality real-time tasks using the static optimization solution [10].

During the run-time, each mixed-criticality task releases jobs every period. The deadline of a high-criticality job released at time t is set as $t+VD_i$, while that of a low-criticality job is $t + P_i$. Those released jobs are inserted into the ready queue for execution. We use EDF (Earliest Deadline First) for scheduling jobs so that the job with the earliest deadline is dispatched from the queue. The frequency controller adjusts the operating CPU frequency depending on the job criticality. In the proposed scheme, we adjust the frequency level of a high-criticality job below f^{HI}. Similarly, the frequency level of a low-criticality job is set below f^{LO} depending on the system utilization.



(a) The proposed DVFS scheduling in low mode



(b) The proposed DVFS scheduling in high mode

Fig. 5. The proposed system

Figure 5(a) shows the related flow of the proposed system in low mode. When a high-criticality job does not finish its low-mode execution time, the mode switch occurs. In high mode, all low-criticality jobs are dropped from the system, as shown in Figure 5(b). The frequency level is set as the maximum CPU frequency. In this paper, we only consider the energy reduction in low mode.

B. DVFS Scheduling Algorithms

In this subsection, we explain the proposed scheme in detail. Table II shows additional notations used in algorithm description. We update the utilization of a high-criticality task at the time of arrival and departure and denote it as U_i. A periodic job of task τ_i is denoted as J_i which is defined by two parameters: execution time and deadline. The execution time of J_i is given by $C_i(LO)$. The deadline of a low-criticality job J_i given by $t + P_i$ when the job is released at time t.

TABLE II. NOTATIONS

Notation	Meaning
U_i	The utilization of task τ_i
J_i	A periodic job of task τ_i
D_i	The absolute deadline of a job J_i
t	The current time
$\mathcal{Q}_{ready}$	The CPU ready queue
J_{curr}	The current job in execution

Since we use EDF-VD scheduling algorithm, the deadline of a high-criticality job J_i is given by VD_i (= $x \cdot P_i$). We denote the deadline of a job J_i as D_i in the algorithm description.

The proposed scheme consists of several functions which are called some events in the system, as shown in Figure 5(a). The followings are a brief description of each function and the pseudo-codes are provided in Figure 6 and Figure 7.

```
1: function RELEASE-JOB(τ_i)
2:     Insert a job J_i(C_i(LO), t + VD_i) into Q_ready
3:     if X_i = LO then                          ▷ low-criticality job
4:         U_i ← (C_i(LO)/f^LO)/P_i
5:     else
6:         U_i ← (C_i(LO)/f^HI)/P_i +
               (C_i(HI) − C_i(LO))/P_i
7:     end if
8:     SCHEUDULE-PMC( )
9: end function

10: function FINISH-JOB(J_i)
11:     if X_i = HI then                          ▷ high-criticality job
12:         if J_i finishes C_i(LO) completely then
13:             U_i ← (Ci(LO)/f^HI)/P_i
14:         else
15:             The system mode is changed from Low to High.
16:         end if
17:     end if
18:     SCHEDULE-PMC( )
19: end function

20: function SCHEDULE-PMC( )
21:     if Q_ready ≠ ∅ then
22:         J_k ← the job with the earliest deadline in Q_ready
23:         if J_curr = ∅ then                    ▷ CPU idle
24:             J_curr ← J_k
25:         else if D_k < D_curr then             ▷ Preemption by EDF
26:             J_curr is preempted and re-inserted into Q_ready
27:             J_curr ← J_k
28:         end if
29:         SELECT-FREQUENCY( )
30:     end if
31: end function
```

Fig. 6. Algorithm of scheduling power-aware mixed-criticality tasks

```
1: function SELECT-FREQUENCY( )
2:     if the system is in High mode then
3:         The frequency is set as f_max.
4:     else                                  ▷ The system is in Low mode.
5:         U ← min(Σ_{i=1}^{n} U_i, 1.0)
6:         if X_curr = LO then
7:             U ← U × f^LO
8:         else
9:             U ← U × f^HI
10:        end if
11:        freq ← the minimum f_i ∈ F s.t. U ≤ f_i/f_max
12:        The frequency is set as freq.
13:    end if
14: end function
```

Fig. 7. Algorithm of selecting frequency

- *Release-Job (τ_i):* This function is called when a task τ_i releases a job every P_i period.
- *Finish-Job (J_i):* When a job is executed completely, the function is called to finish the job.
- *Schedule-PMC ():* This function schedules jobs in the ready queue, $\mathcal{Q}_{ready}$, at the time of job arrival or departure.
- *Select-Frequency ():* When a job is assigned to the CPU, the frequency level is determined by this function.

At the time of job arrival, the job is inserted in the ready queue. In addition, we update the utilization of each task. Since the frequency of a low-criticality task is given by f^{LO}, the task utilization is determined by the equation in line 4 of Figure 6. In case of high-criticality task, there is additional computation time of $C_i(HI) - C_i(LO)$ every period so that the utilization is given by the equation in line 6. After updating the task utilization, we call the scheduling algorithm (line 8).

When a job J_i finishes its computation completely, the function *Finish-Job* is called. If the job is high-criticality and requires more execution, the system mode is changed from low to high (line 15 of Figure 6). Otherwise, we update the high-criticality task utilization as in the equation of line 13. Whenever a high-criticality job completes its low-mode execution, the additional reservation time of $(C_i(HI) - C_i(LO))$ is not required in that period.

The function *Schedule-PMC ()* schedules jobs in the ready queue by EDF (line 22-28 of Figure 6). At the end of each scheduling event, we call *Select-Frequency ()* function in order to adjust the CPU frequency dynamically. As shown in Figure 7, the utilization is calculated first (line 5). The frequency level is selected by the least frequency level such that the relative speed is greater than or equal to the required system utilization. Since the selected frequency level satisfies the required utilization, the schedulability is guaranteed.

C. Example

Let us consider the task set in Table I as an example. As discussed in Section III.C, the optimal x, f^{HI}, and f^{LO} are

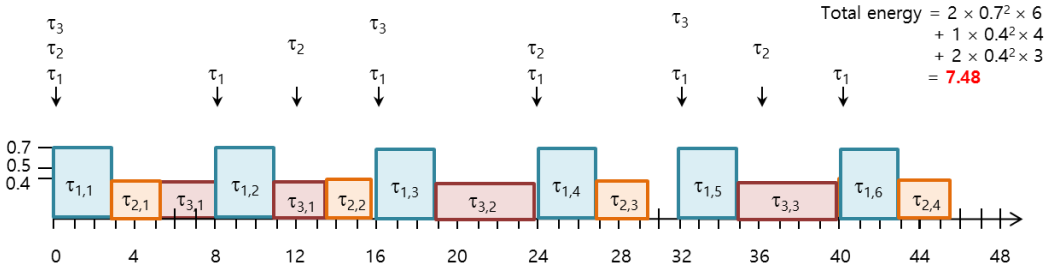


Fig. 8. Scheduling results of mixed-criticality real-time tasks

given by 0.625, 0.7, and 0.5, respectively. Figure 8 shows the scheduling results of the proposed scheme. All jobs are scheduled by EDF with DVFS, as shown in Figure 8.

When task τ_1 releases job periodically, it is scheduled first due to the earliest deadline. After task τ_1 finishes its low-mode execution time, the task utilization is decreased by the algorithm (line 13 of Figure 6). Therefore, two low-criticality tasks τ_2 and τ_3 can be slow down more than the given frequency level f^{LO} = 0.5. The total energy in Figure 8 is given by 7.48 which reduces about 10% more than that of the previous static scheme in Figure 4.

V. Simulation Results

A. Simulation Environment

In the simulations, we assume seven discrete frequency levels as shown in Table III. The execution time of a task in low mode is generated randomly from 1 to 5. And then, the period is derived so as to be the target utilization. We fix the utilization of low-criticality tasks is 0.2, while the utilization of high-criticality tasks in low mode is varied from 0.2 to 0.4. The execution time of a high-criticality task in high mode is defined by a *low-to-high* rate, denoted as r, which is varied from 1.5 to 2.5. That is, $C_i(HI)$ of a high-criticality task becomes $r \times C_i(LO)$.

The number of tasks in a task set is four, two with low-criticality and two with high-criticality. We generate 100 random task sets for each simulation and measure the average power consumption. For a given task set, we simulate running the task set for the least common multiple of the tasks' periods.

TABLE III. Simulation Parameters

Parameters	Values
Frequency levels	{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 }
Utilization of low-criticality tasks (U_{LO}^{LO})	0.2
Utilization of low-mode high-criticality tasks (U_{LO}^{HI})	0.2, 0.3, 0.4
Low-to-high rate (r)	1.5, 2.0, 2.5, 3.0
Number of low-criticality tasks	2
Number of high-criticality tasks	2

B. Impact of the rate of low to high mode jobs in high-criticality tasks

Figure 9 shows the average energy consumption for various rates of low to high mode jobs. We fix U_{LO}^{HI} as 0.3. We compare the proposed scheme with the previous work in [10]. As shown in Figure 9, as the rate increases, the performance becomes much. The proposed scheme decreases the frequency level by using reservation time scheduled for high-criticality tasks. Thus, as $C_i(HI)/C_i(LO)$ is increased, the slack time is also increased, which results in low energy in the proposed scheme.

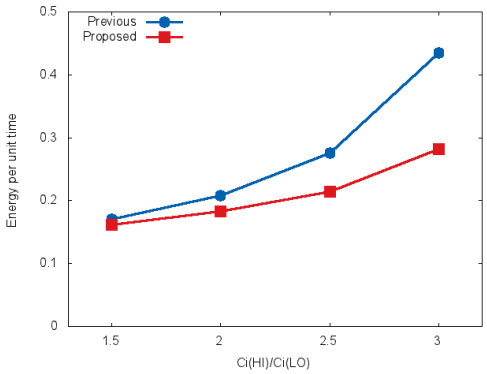


Fig. 9. Impact of rates of low to high mode jobs in high-criticality tasks ($U_{LO}^{HI} = 0.3$)

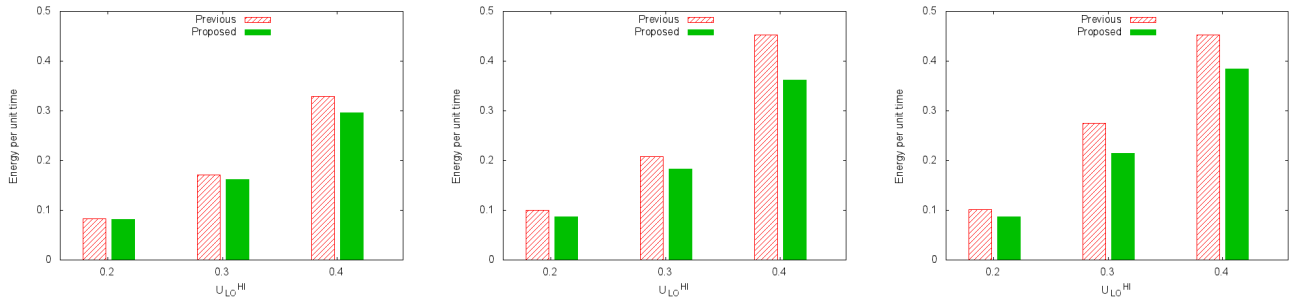
C. Impact of utilization of high tasks

Figure 10 shows simulation results for various utilizations of high-criticality tasks. As the utilization of high-criticality tasks in low mode increases, the possibility of slack time is also increased. Thus, the energy reduction in high U_{LO}^{HI} becomes larger.

D. Analysis of optimal frequency levels

We also analyze whether optimal frequency levels of the proposed scheme differ from those of the static scheme in [12]. For example, Figure 11 shows unit energy consumption for various x of the example in Table I. The optimal x in the previous static scheme is 0.625 as indicated in Figure 3, while that in the proposed dynamic scheme is different, as shown in Figure 11. That is, the optimal frequency levels derived in the static scheme may not be optimal in the dynamic scheme.

Table IV shows how many such cases occur in the simulation results, where $U_{LO}^{HI} = 0.3$ and $C_i(HI)/C_i(LO) = 2.0$. As indicated in Table IV, the proposed scheme shows that the optimal frequency levels for the static scheme do not guarantee the optimal frequency levels in dynamic schemes.



(a) $C_i(HI)/C_i(LO) = 1.5$ (b) $C_i(HI)/C_i(LO) = 2.0$ (c) $C_i(HI)/C_i(LO) = 2.5$

Fig. 10. Impact of rates of utilization of high tasks

TABLE IV. THE NUMBER CASES WITH DIFFERENT OPTIMAL FREQUENCY LEVELS

$C_i(HI)/C_i(LO)$	U_{LO}^{HI}		
	0.2	0.3	0.4
1.5	0	11	30
2.0	42	52	100
2.5	27	80	100

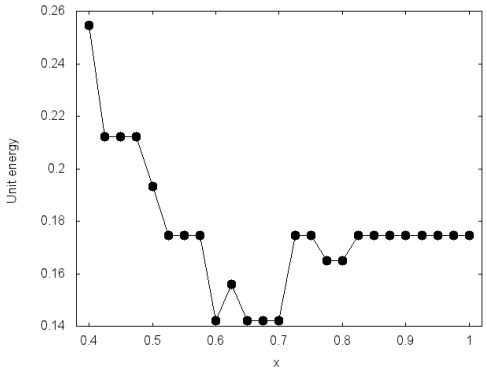


Fig. 11. The unit energy for various x in the proposed scheme

VI. CONCLUSIONS

In this paper, we provide a new dynamic power-aware scheduling scheme of mixed-criticality real-time tasks under discrete frequency levels. The proposed scheme adjusts the CPU frequency levels of high-criticality and low-criticality tasks below the frequencies derived from the static scheme. The proposed scheme uses slack times reserved by high-criticality tasks. Throughout simulation results, we showed that the proposed scheme reduces more energy and the optimal frequency levels in the static scheme may not be optimal in the dynamic scheme.

We are currently focusing on more analysis of power-aware scheduling including the proposed scheme. For example, we can improve the proposed dynamic DVFS scheme by using slack time analysis. In addition, we plan to derive optimal frequency levels in the dynamic scheme, and implement the proposed scheme in RTLinux-based systems.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. NRF-2015R1A1A1A05001369) and BK21+ program by the Ministry of Education.

REFERENCES

[1] Avionics application software standard interface: Part 1. Technical report, Avionics Electronic Engineering Committee (ARINC), 2006.

[2] A. Burns and R. I. Davis. Mixed criticality systems - A review. Technical report, University of York, http://www-users.cs.york.ac.uk/burns/review.pdf.

[3] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proc. of 28th IEEE Real-Time Systems Symp.*, 2007.

[4] S. Baruah, A. Burns, and R. Davis. Response-time analysis for mixed criticality systems. In *Proc. of 32nd IEEE Real-Time Systems Symp.*, 2011.

[5] S. Baruah, V. Bonifaci, G. D. Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie. Scheduling real-time mixed-criticality jobs. *IEEE Transactions on Computers*, vol. 61, pp. 1140-1152, August 2012.

[6] H. Li and S. Baruah. An algorithm for scheduling certifiable mixed-criticality sporadic task systems. In *Proc. of the 31st IEEE Real-Time Systems Symposium (RTSS)*, 2010.

[7] N. Guan, P. Ekberg, M. Stigge, and W. Yi. Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems. In *Proc. of 32nd IEEE Real-Time Systems Symp.*, 2011.

[8] H. Aydin, R. Melhem, D. Mosse, and P. M-.Alvarez. Power-aware scheduling poer periodic real-time tasks. *IEEE Trans. on Computers*, vol. 53, no. 5, pp. 584-600, May 2004.

[9] S. Baruah and Z. Guo. Mixed-criticality scheduling upon varying-speed processors. In *Proc. of 34th Real-Time Systems Symposium*, December 2013.

[10] P. Huang, P. Kumar, G. Giannopoulou, and L. Thiele. Energy efficiient DVFS scheduling for mixed-criticality systems. In *Proc. of the 14th International Conference on Embedded Software (EMSOFT '14)*, October 2014.

[11] F. Dorin, P. Richard, M. Richard, and J. Goossens. Schedulability and sensitivity analysis of multiple criticality tasks with fixed-priorities. *Real-Time Systems*, vol. 46, pp. 305-331, December 2010.

[12] S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable mixed-criticality systems. In *Proc. of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2010.

[13] N. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. University of York, England, Tech. Rep., 1991.

[14] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Proc. of the 13th Euromicro Conference on Real-Time Systems*, pp. 225–232, 2001.

[15] F. Gruian, Hard real-time scheduling for low-energy using stochastic data and dvs processors. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, 2001.

[16] R. Jejurikar and R. Gupta. Optimized slowdown in real-time task systems. *IEEE Trans. Computers*, vol. 55, no. 12, pp. 1588–1598, December 2006.

[17] X. Zhong and C.-Z. Xu. System-wide energy minimization for real-time tasks: Lower bound and approximation. *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 28:1–28:24, May 2008.

[18] H. Yun, P.-L. Wu, A. Arya, C. Kim, T. Abdelzaher, and L. Sha. System-wide energy optimization for multiple dvs components and real-time tasks. *Real-Time Systems*, vol. 47, no. 5, pp. 489–515. 2011.

[19] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, vol. 53, no. 5, pp. 584–600, 2004.

[20] P. Mejia-Alvarez, E. Levner, and D. Mossé. Adaptive scheduling server for power-aware real-time tasks. *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 2, pp. 284–306, May 2004.

[21] M. Marinoni and G. Buttazzo. Elastic dvs management in processors with discrete voltage/frequency modes. *IEEE Transactions on Industrial Informatics*, vol. 3, no. 1, pp. 51–62, February 2007.

[22] W. Wang, S. Ranka, and P. Mishra. Energy-aware dynamic slack allocation for real-time multitasking systems. *Sustainable Computing: Informatics and Systems*, vol. 2, no. 3, pp. 128–137, 2012.

[23] J. Mei, K. Li, J. Hu, S. Yin, and E. H.-M. Sha. Energy-aware preemptive scheduling algorithm for sporadic tasks on dvs platform. *Microprocessors and Microsystems*, vol. 37, no. 1, pp. 99–112, 2013.

[24] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. *SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 89–102, October 2001.

[25] S. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie. Mixed-criticality scheduling of sporadic task systems. In *Proc. of the 19th European Conf. on Algorithms*, 2011.

[26] L. Niu and G. Quan. Reducing both dynamic and leckage energy consumption for hard real-time systems. In *Proc. of the International Conference on Compilers, Architectures and Systhesis for Embedded Systems (CASES 2004)*, Washington, DC, USA, September 2004.