DVFS Virtualization for Energy Minimization of Mixed-Criticality Dual-OS Platforms

Takumi Komori*, Yutaka Masuda*, and Tohru Ishihara*
*Graduate School of Informatics, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, JAPAN

***Abstract*—A dual-OS platform can efficiently implement emerging mixed-criticality systems by consolidating a real-time OS (RTOS) and a general-purpose OS (GPOS). Although the dual-OS platform attracts increasing attention, it often suffers from energy inefficiency in the GPOS for guaranteeing real-time responses of the RTOS. This paper proposes an energy minimization method called DVFS virtualization, which allows running multiple DVFS policies dedicated to the RTOS and GPOS, respectively. The experimental evaluation using a commercial processor showed that the proposed hardware could change the supply voltage within 500 ns and reduce the energy consumption of typical applications by 60 % in the best case compared to conventional dual-OS platforms.**

***Index Terms*—Dynamic voltage scaling, Virtual machine monitors, Real-time systems, Mixed-criticality, Embedded software**

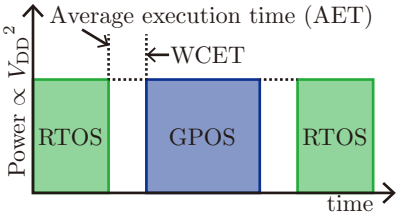
I. Introduction

Recent embedded systems are becoming complicated day by day. For example, robots now have to capture images from a camera, process them by OpenCV, and interact with an operator remotely via an SSL-protected connection; while controlling machinery as traditional. A mixed-criticality system is this kind of embedded system which serves both critical machinery control and non-critical information processing.

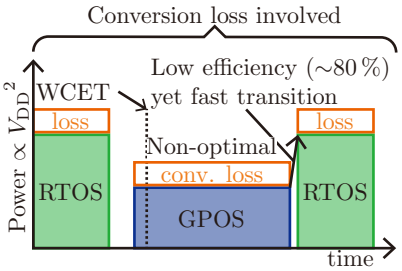
A dual-OS platform is suitable for implementing the mixed-criticality system [1]–[6]. A lightweight real-time OS (RTOS) is tiny and verifiable, bringing highly reliable real-time capability desirable for machinery control. However, the RTOS with limited functionality is unsuitable for implementing a complicated large-scale application. In contrast, a general-purpose OS (GPOS) like Linux has rich libraries and utilities. Moreover, a developer can use open-source software (OSS) without modification. Therefore, the GPOS provides a highly efficient development environment. Unfortunately, constant updates of the GPOS imply that complete validation of such a colossal software is nearly impossible. In addition, an unmodified commercially available GPOS cannot guarantee real-time capability due to its significant interrupt latency and difficulty in predicting the behavior [7]–[9]. The dual-OS platform takes advantage of the RTOS and GPOS by simultaneously executing the OSes with virtualization.

Implementing the dual-OS platform is widely studied; however, no previous research tackled minimizing energy consumption. As we explain later, existing energy reduction methods for mixed-criticality systems rely on a single scheduler and cannot be applied to the dual-OS platform.

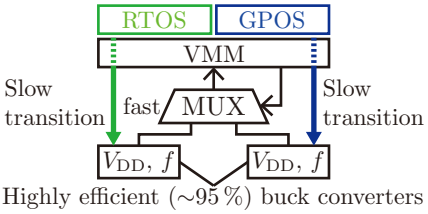
This paper, for the first time, proposes an energy minimization method for the dual-OS platform called DVFS virtualization. Figure 1 summarizes our method. Conventional dual-OS platforms shown in Fig. 1(a) must keep the performance

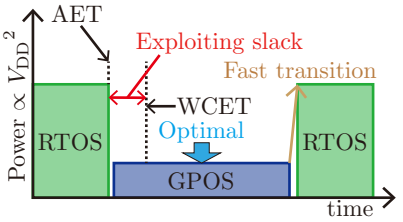


(a) Conventional DVFS



(b) Conventional DVFS with fast transition regulator





(c) Proposed DVFS virtualization

Fig. 1: Summary of the proposed energy minimization. Existing dual-OS platforms must keep the performance high enough for the RTOS and suffer from low energy efficiency (a). A voltage regulator capable of changing its output fast enables the performance lowering of the GPOS, but an on-chip fast regulator is inefficient and can only produce a few levels of voltage (b). The proposed method realizes fast performance transition using two highly-efficient voltage regulators and schedules the GPOS based on the average execution time of the RTOS. The GPOS can aggressively lower its performance for the dynamic slack induced by the RTOS, improving the energy efficiency of the dual-OS platform (c).

high enough, and the GPOS cannot aggressively reduce energy consumption. Regulators capable of changing output voltage fast allow the GPOS to run slower but suffer from energy loss due to the low conversion efficiency [10], [11], as shown in Fig. 1(b). More detail on the relation between the conversion efficiency and the transition speed in voltage regulators is summarized in Sect. IV. Our method provides stable power supplies with a high conversion efficiency of 95 % at the best case [12] and clocks dedicated to both OSes. The power supplies and clocks are quickly switched by multiplexers, as shown in Fig. 1(c). As a result, like OSes are virtualized, the RTOS and GPOS can adjust their performance independently to reduce the energy consumption of the dual-OS platform efficiently. Since the behavior of both the OSes is monitored by the virtual machine monitor (VMM) depicted in Fig. 1(c), the GPOS can exploit the slack dynamically produced by the early completion of the RTOS to reduce more energy. The detailed algorithm for exploiting the dynamic slack to minimize energy consumption is explained in Sect. III-A. The contribution of the paper is summarized as follows;

- DVFS algorithms to minimize the energy consumption of the dual-OS platform are proposed for the first time. The algorithms reduce energy consumption efficiently by applying different optimal DVFS policies for each OS.
- A fast transition mechanism of voltage and frequency is also proposed. Without this mechanism, the proposed DVFS algorithm might violate the real-time constraints of the RTOS.

Combining the above proposed DVFS algorithms and the fast transition mechanism for voltage and frequency realizes DVFS virtualization, which carries out individual DVFS for each OS. The speed and energy overhead of DVFS virtualization is verified negligible.

The rest of the paper is constructed as follows: Section II shows the execution model of a typical dual-OS platform and the downside of the traditional energy minimization method for a mixed-criticality system. Section III proposes DVFS algorithms to minimize the energy of the dual-OS platform and shows the difficulty of realizing it. Section IV proposes hardware implementation of DVFS virtualization. Section V evaluates the proposed method using a commercial processor. Section VI concludes the paper.

II. Related Works

This section introduces the execution model of a general dual-OS platform and existing energy minimization algorithms for mixed-criticality systems. We also describe the issue of applying existing algorithms to the dual-OS platform. Additionally, we later summarize existing voltage regulators in Sect. IV.

A. Execution model of dual-OS platform

A general virtualized dual-OS platform consolidates an RTOS and a GPOS, as shown in Fig. 2. The processor has three privilege levels: User, Supervisor, and Hypervisor modes. The VMM runs on the Hypervisor mode to manage virtual CPU

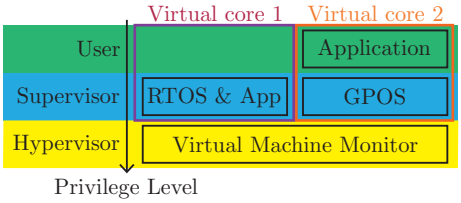


Fig. 2: The example of dual-OS consolidation.

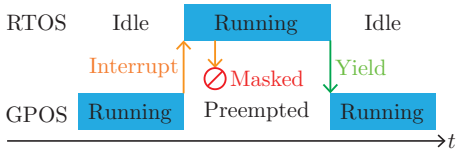


Fig. 3: Dual-OS execution scheme.

cores. Under the VMM, the RTOS and GPOS run as illustrated in Fig. 3. The GPOS can only run when the RTOS is idle. The processor executes the RTOS immediately on RTOS interrupts anytime during the processor executes the GPOS. All the GPOS interrupts are disabled while the RTOS is running. After the RTOS completes all the tasks and becomes idle again, the GPOS resumes. The RTOS always takes precedence over the GPOS, and it guarantees real-time capability.

B. Existing mixed-criticality DVFS algorithms

Dynamic Voltage and Frequency Scaling (DVFS) is one of the most popular methods to minimize the energy consumption of the processor, which can drastically reduce energy consumption by lowering operating voltage and performance (clock frequency). When applying DVFS to the embedded system, lowering the performance as much as possible while guaranteeing real-time constraints is most important. Therefore, algorithms to keep the performance of the processor minimum required, especially methods to incorporate DVFS into the OS, have been widely investigated over the years [13].

The most popular methods to implement the mixed-criticality system are the extended scheduling theory inherited from [14]. Like the traditional scheduling theory, such methods schedule tasks based on minimum activation periods and worst-case execution times (WCETs). We refer to this type of mixed-criticality system as a single-OS platform since one scheduler handles all the tasks regardless of their criticality.

Previous DVFS algorithms for the single-OS platform model all the tasks regardless of their criticality and reduce energy consumption while guaranteeing real-time constraints [15]–[18]. Since schedulability and energy consumption are trade-offs, parameters must be optimized to balance them. Scheduling with non-optimal parameters causes an increase in energy consumption by around 20 % [15].

Single-OS platforms are mainly developed for automobiles and avionics, where all the tasks are strictly modeled. Thus, algorithms for the single-OS platform cannot be applied to embedded systems requiring the GPOS due to the following reasons:

1) One cannot determine the minimum activation periods and WCETs of tasks. It is unrealistic to analyze a colossal GPOS and OSS running on it and determine the minimum activation periods and WCETs of every process and thread.
2) Scheduling algorithms are different. Single-OS platforms assume priority-based scheduling, while the GPOS uses round-robin-based scheduling.
3) The latency of switching the supply voltage and clock frequency is rarely considered. General buck converters and PLLs take roughly $100\,\mu s$ to change their output and might affect real-time capability [13]. Most previous works evaluate their algorithms only in simulations [15]–[18], showing that applying DVFS for the single-OS platform on a physical processor is unrealistic.

The above discussions show that embedded systems requiring the GPOS cannot utilize energy minimization algorithms for single-OS platforms.

A new DVFS algorithm applicable to the dual-OS-based mixed-criticality system is therefore needed. The dual-OS platform drastically improves software flexibility by allowing the use of the GPOS. Essential OSS such as ROS, OpenCV, GStreamer, and others does not operate on any system other than the GPOS. Customers can add, remove, and update applications freely on the GPOS while protecting the static supplier application on the RTOS. The dual-OS platform is essential to consolidate real-time operations and functions requiring the GPOS, and its energy minimization is desired.

III. Energy minimization of dual-OS platform

This section proposes several DVFS algorithms to minimize the energy consumption of the dual-OS platform under different practical situations, respectively. The algorithms can be divided roughly into two groups whether they offer the soft real-time capability to the GPOS (Sect. III-A2) or not (Sect. III-A1). Furthermore, for the soft real-time GPOS, we propose a method to exploit the early completion of the RTOS (Sect. III-A3) and an extension for a processor which only supports a discrete set of frequencies (Sect. III-A4). Additionally, this section discusses the difficulty of implementing the algorithm in Sect. III-B. The energy reduction thanks to the proposed algorithms are later evaluated in Sect. V.

Note that the operating voltage is also scaled to the appropriate value corresponding to the frequency as is done in the traditional DVFS techniques, although only frequency scaling is explicitly described in the following explanation. The performance of the GPOS must be determined based on its CPU utilization. The following example controls the clock frequency of the GPOS for 10 unit times (except for Fig. 4). The GPOS runs at the same clock frequency within this time period when running on a processor that supports the continuous clock frequency. For a processor that only supports a discrete set of frequencies, the GPOS changes its clock frequency once in the time period.

A. DVFS algorithms for dual-OS platform

1) Naive approach: The most naive DVFS algorithm for the dual-OS platform is letting the RTOS run at the fastest and the GPOS at the slowest clock frequency, as shown in Fig. 4. If no DVFS is applied, the RTOS executes two tasks, T_1 and T_2, at the maximum clock frequency $f_{\max}$, and the GPOS is executed at $f_{\max}$. The sum of the CPU utilization of the RTOS and GPOS is less than 1, and idle time exists. If DVFS is applied, the clock frequency of the GPOS is set to the minimum ($f_{\min}$), while T_1 and T_2 remain to run at $f_{\max}$, reducing the energy consumed by the GPOS. However, the execution of the GPOS does not complete, and the idle time is eliminated. Assuming the supply voltage and clock frequency are scaled fast enough, this case is equivalent to continuing to operate the processor at the fastest frequency for the RTOS. Since every task of the RTOS should meet the deadline at least for the fastest frequency, this algorithm can guarantee the real-time constraint of the RTOS. On the other hand, scheduling of the GPOS is performed as best effort, and no performance is guaranteed. When no soft real-time task exists in the GPOS, this algorithm is easiest to implement, and a considerable energy reduction is expected.

2) Soft real-time approach: Controlling the sum of the CPU usage of the RTOS U_{RT} and GPOS U_{GP} to $U_{\mathrm{RT}} + U_{\mathrm{GP}} = 1$ gives the soft real-time capability to the GPOS, as shown in Fig. 5. Before applying DVFS in Fig. 4, CPU utilization of each OS is $U_{\mathrm{RT}} = 0.5$ and $U_{\mathrm{GP}} = 0.3$. Providing the clock frequency $f_{\mathrm{GP}} = U_{\mathrm{GP}} f_{\max}/(1 - U_{\mathrm{RT}}) = 0.6 f_{\max}$ for the GPOS reduces energy consumption while keeping the performance of the GPOS. Typically, periodically measuring the execution time of the GPOS estimates U_{GP} like CPUFreq governor of Linux [19]. One can also estimate U_{GP} in a more

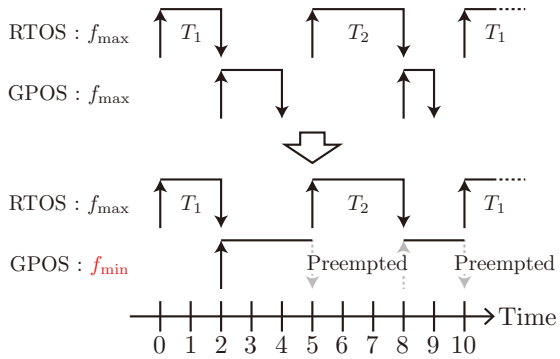


Fig. 4: The naive DVFS algorithm for the dual-OS platform.

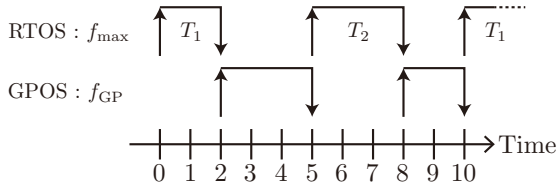


Fig. 5: A DVFS algorithm to give the GPOS soft real-time capability. The GPOS completes its execution unlike Fig. 4.

refined manner [20].

The total energy consumption of the whole circuit operating with the clock frequency f is composed of the following.

- The energy consumption, which is proportional to the square of the supply voltage of the core (the dynamic energy consumption of the core) : $E_{\rm d}(f)$
- The energy consumption, which is proportional to the supply voltage of the core (the static energy consumption of the core) : $E_{\rm s}(f)$
- The power consumption, which is not related to the supply voltage of the core : $P_{\rm s}$

Therefore, denoting the supply voltage corresponding to the clock frequency f as $V_{\rm DD}(f)$, we get the average energy consumption per unit time before and after applying DVFS as described in (1) and (2), respectively.

$$E_0/T = (U_{\rm RT}+U_{\rm GP})f_{\max}\{E_{\rm d}(f_{\max})+E_{\rm s}(f_{\max})\} + (1-U_{\rm RT}-U_{\rm GP})f_{\max}E_{\rm s}(f_{\max})+P_{\rm s}. \quad (1)$$

$$E_{\rm dvfs}/T = U_{\rm RT}f_{\max}\{E_{\rm d}(f_{\max})+E_{\rm s}(f_{\max})\} + U_{\rm GP}f_{\max}\left\{\frac{V_{\rm DD}(f_{\rm GP})}{V_{\rm DD}(f_{\max})}\right\}^2 E_{\rm d}(f_{\max}) + U_{\rm GP}f_{\max}\frac{V_{\rm DD}(f_{\rm GP})}{V_{\rm DD}(f_{\max})}E_{\rm s}(f_{\max})+P_{\rm s}. \quad (2)$$

This algorithm is expected to reduce the dynamic energy consumption of the GPOS proportional to ${U_{\rm GP}}^3$ under the approximation that $V_{\rm DD}(f)\propto f$.

3) Exploiting early completion of RTOS: The CPU utilization of the RTOS is also estimated by periodically measuring the execution time like the GPOS. The periodic measuring offers the average CPU utilization based on the past execution results. However, if the WCETs and the minimum activation periods are given for every task of the RTOS, $U_{\rm RT}$ can be estimated in a more fine-grained manner. For the RTOS, the worst-case CPU utilization is determined from the sum of the worst-case execution times in the hyper-period. On every task completion, the estimated CPU utilization of the RTOS is adjusted lower because the RTOS sometimes completes earlier than its WCET. Since the GPOS is executed as the lowest priority task, modifying the clock frequency for the GPOS when the RTOS enters idle exploits the early completion of the RTOS task in a fine-grained manner. For example, consider two RTOS tasks, T_1 and T_2, whose WCETs are as shown in Fig. 5, and they finish earlier than the WCETs, as shown in Fig. 6. With WCETs, $U_{\rm RT}=0.5$, and $f_{\rm GP}=0.6f_{\max}$. When T_1 finishes at the time 1, the worst-case $U_{\rm RT}$ is updated to 0.4. Therefore, after the finish of T_1, the GPOS can run with $f_{\rm GP1}=0.5f_{\max}$. In addition, after T_2 finishes, $U_{\rm RT}$ is updated to 0.3, determining $f_{\rm GP2}=f_{\max}/3$.

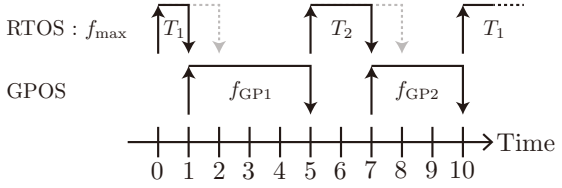


Fig. 6: Exploiting the early completion of the RTOS.

4) When processor can only operate with discrete set of frequencies: General processors can only operate with a discrete set of clock frequencies, and the GPOS cannot run at $f_{\rm GP}$. To achieve the maximum energy efficiency, the GPOS should run using two different clock frequencies of $f_{\rm GPH}$ and $f_{\rm GPL}$ ($f_{\rm GPH}>f_{\rm GP}>f_{\rm GPL}$) so that the GPOS runs at $f_{\rm GP}$ on average, as shown in Fig. 7. Selecting the frequency closest to $f_{\rm GP}$ as $f_{\rm GPH}$ and $f_{\rm GPL}$ minimizes energy consumption.

In this case, the GPOS consumes $U_{\rm GPL}$ of the CPU time at $f_{\rm GPL}$ and $U_{\rm GPH}$ of the time at $f_{\rm GPH}$. Here,

$$U_{\rm GPL}=\frac{(1-U_{\rm RT})(f_{\rm GPH}-f_{\rm GP,AVE})}{f_{\rm GPH}-f_{\rm GPL}}, \quad (3)$$

and $U_{\rm GPH}=1-U_{\rm RT}-U_{\rm GPL}$. The total CPU utilization is $U_{\rm RT}+U_{\rm GPH}+U_{\rm GPL}=1.0$, meaning no idle time exists. Note that the number of instructions the GPOS executes for the unit time is kept equal, and the performances of the GPOS before and after applying DVFS are equivalent. The GPOS is recommended to begin with $f_{\rm GPL}$ at the start of a time period and switch to $f_{\rm GPH}$ later. Starting with $f_{\rm GPL}$ helps the GPOS run at the slower clock frequency for a longer time when the algorithm discussed in Sect. III-A3 gradually lowers $U_{\rm RT}$ (and $U_{\rm GPH}$) on task completion.

With the discrete operating conditions, average energy consumption per unit time is expressed as

$$E_{\rm discrete}/T = U_{\rm RT}f_{\max}\{E_{\rm d}(f_{\max})+E_{\rm s}(f_{\max})\} + U_{\rm GPH}f_{\rm GPH}\left\{\frac{V_{\rm DD}(f_{\rm GPH})}{V_{\rm DD}(f_{\max})}\right\}^2 E_{\rm d}(f_{\max}) + U_{\rm GPH}f_{\rm GPH}\frac{V_{\rm DD}(f_{\rm GPH})}{V_{\rm DD}(f_{\max})}E_{\rm s}(f_{\max}) + U_{\rm GPL}f_{\rm GPL}\left\{\frac{V_{\rm DD}(f_{\rm GPL})}{V_{\rm DD}(f_{\max})}\right\}^2 E_{\rm d}(f_{\max}) + U_{\rm GPL}f_{\rm GPL}\frac{V_{\rm DD}(f_{\rm GPL})}{V_{\rm DD}(f_{\max})}E_{\rm s}(f_{\max}) + U_{\rm idle}f_{\min}\frac{V_{\rm DD}(f_{\min})}{V_{\rm DD}(f_{\max})}E_{\rm s}(f_{\max})+P_{\rm s}. \quad (4)$$

Note that if $f_{\rm GP}<f_{\min}$, we cannot eliminate the idle time and let the GPOS run at $f_{\min}$ for $U_{\rm GP}f_{\max}/f_{\min}$ of CPU time.

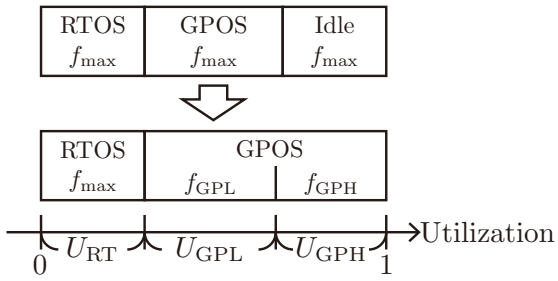


Fig. 7: A DVFS scheme for the processor with discrete set of clock frequencies.

In this case, $U_{\mathrm{idle}} = 1 - U_{\mathrm{RT}} - (f_{\max}/f_{\min})U_{\mathrm{GP}}$ of CPU time is consumed by the processor idling.

B. Difficulty of implementing algorithms

In conventional dual-OS platforms, the GPOS only runs when the RTOS is idle, and the RTOS can preempt the GPOS anytime [1]–[6]; therefore, generally, one cannot predict when the RTOS starts executing and requires high performance. When the RTOS interrupts while the GPOS is running slower, the RTOS continues to run at the same low performance and might violate real-time constraints because general DC-DC converters and PLLs take roughly 100 µs to change their output [13]. As discussed above, the proposed DVFS algorithms cannot be applied to the conventional dual-OS platform as is because they might violate real-time constraints. Generally, in the conventional dual-OS platform, the GPOS must be conservative to guarantee real-time constraints of the RTOS, and the energy reduction becomes insufficient.

IV. Implementation of DVFS Virtualization

This section shows the hardware and software structure to realize DVFS algorithms discussed in Sect. III-A.

A. Hardware component for DVFS virtualization

This section proposes the hardware structure shown in Fig. 8, which realizes immediate performance change and energy minimization in the dual-OS platform. Usually, the processor is equipped with only one PLL for its core clock. However, we use two PLLs dedicated to the RTOS and GPOS, and we also use different power supplies for them. Additionally, an analog multiplexer is used to switch the power supply quickly.

With the proposed hardware, the RTOS and GPOS have their power supplies and clocks to perform DVFS control without interfering with each other; we refer to this DVFS virtualization. From the software's perspective, the proposed hardware is seen as illustrated in Fig. 9. Two virtual processors dedicated to the RTOS and GPOS are running on the same physical processor, and they run with independent power supplies and clock frequencies. The RTOS wakes up immediately after interrupting the GPOS execution like the present virtualization methods. At the same time, the proposed hardware also

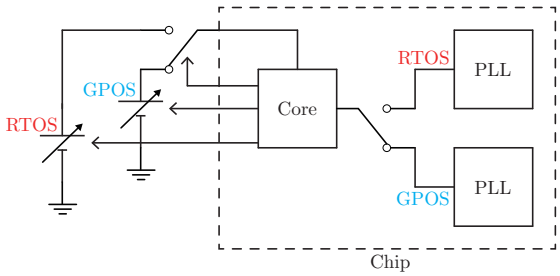


Fig. 8: Proposed hardware structure. The RTOS and the GPOS have their own power supplies and regulators. The multiplexer changes the performance immediately along with an OS switching.

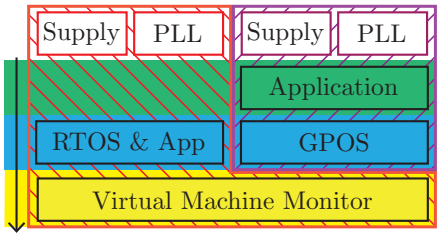


Fig. 9: The proposed system from the software's perspective.

switches the power supply and the clock frequency. Therefore, no matter how slow the GPOS is, the RTOS can resume to a high-speed mode instantly. Virtualization in software means a technique to pretend simultaneous execution of multiple OSes by switching them quickly. The proposed hardware pretends simultaneous execution of multiple DVFS algorithms by switching the supply and clock instantly; this helps realize the proposed DVFS algorithms, which use different DVFS policies for each OS.

Previous studies proposed a DVFS method called voltage dithering, which uses a small number of stable supply voltages and switches them instantaneously to meet the speed of the processor to the speed required [21]. However, our method is different from the dithering since our method is possible to use fine-grained programmable voltage sources with high conversion efficiency by allowing a slow transition of voltage supply dedicated to individual OSes. Voltage dithering can only perform coarse-grained DVFS since the rapid transition between a small number of discrete voltages is more important than fine-grained DVFS for the dithering. In contrast, our method can realize more energy-efficient DVFS using fine-grained supply voltage.

Another previous study proposed an on-chip buck converter using an integrated inductor that realizes voltage transition within 20 ns [10]; however, integrating an inductor is quite difficult, resulting in only around 70 % conversion efficiency. An on-chip, 20 ns transition time switched-capacitor converter with integrated capacitors instead of an inductor is also proposed [11]. Nevertheless, switched-capacitor can only output discrete voltages like dithering and cannot achieve conversion efficiency higher than 90 %, which is inferior to off-chip buck converters. In addition, on-chip LDO can respond fast enough but suffers from a low conversion efficiency lower than both buck and switched capacitor converters [10], [11]. Our method can utilize inductor-based buck converters, which provide a conversion efficiency of more than 95 % in the best case [12]. This type of converter offers slow-transition yet fine-grained voltage levels to the processors, which enables the processor to run at the optimal supply voltage.

In summary, an ideal processor for the proposed method should be equipped with the following features.

- It can execute an RTOS and GPOS simultaneously by a virtualization support mechanism.
- It has two PLLs to generate the core clock and can switch them immediately by a synchronized multiplexer.

- It can change the supply voltage and clock frequency along with an OS switching.
- It can operate with continuously changing supply voltage and corresponding clock frequency.

B. Software component for DVFS virtualization

A general dual-OS platform comprises the VMM, RTOS, and GPOS. Each component has the following suitability for implementing DVFS virtualization.

- The VMM is the most suitable component to implement DVFS virtualization because it can measure CPU utilization and switch the supply and clock along with the OS switching. However, the VMM must somehow know the execution state of tasks of the RTOS to exploit their early completion, which requires a communication channel between the VMM and RTOS.
- The RTOS is suitable for implementing DVFS virtualization. A general RTOS supports cyclic tasks, which is useful to measure CPU utilization periodically. Additionally, the execution states of tasks can be handled, and the early completion is easily exploited. Nevertheless, the supply and clock switching at the preemption should not be implemented in the RTOS. Suppose the situation that the RTOS interrupts when the GPOS is running slowly, and the RTOS has to immediately start running faster than the GPOS to meet the deadline. The VMM handles the interrupt first, saves the context of the GPOS, and distributes the interrupt to the RTOS. Therefore, if the supply and clock are controlled after the RTOS starts, the acceleration of the clock might delay and affect real-time capability. Note that the supply and clock switching at the transition from the RTOS to GPOS can safely be implemented in the RTOS.
- The GPOS should not implement DVFS virtualization. The GPOS handling critical hardware such as the power supply and clock might lead to a fatal situation. For example, the GPOS might incorrectly control the supply voltage and prevent the correct operation of the RTOS because of a bug. Furthermore, multiple research warns that the GPOS can illegally access the information of the Secure side of ARM TrustZone —the most popular virtualization mechanism for the dual-OS platform— by maliciously controlling DVFS [22], [23].

As discussed above, the VMM and RTOS are suitable for implementing DVFS virtualization. Implementing only supply and clock switching on the VMM and others on the RTOS is also recommended.

V. Experimental Results

This section provides experimental results of voltage transition latency and energy efficiency of the proposed method using a commercial processor board integrating ARM Cortex-M33. Section V-A describes the hardware structure used for the experiment. Section V-B shows that even if the GPOS runs slowly, the RTOS can start to run fast within 500 nanoseconds. Section V-C explains how the proposed method is emulated using the commercial processor. Section V-D shows that the energy consumption according to (4) can be obtained by the proposed method, and the extra energy consumption introduced by adding hardware components is negligible. Section V-E shows that the proposed method reduces the energy consumption of typical applications by 60 % in the best case.

A. Evaluation setup

This paper selects NXP i.MX RT685 as a target microcontroller unit (MCU) to emulate the proposed platform. As discussed in Sect. IV, the ideal processor to implement the proposed method is equipped with specific features. Unfortunately, no processor on the current market can be treated as ideal. Although RT685 is an embedded MCU and not an application processor, simple core and peripheral configurations make it easy to implement software. In addition, the evaluation board is designed to allow access to the power supply of the core, and we can easily experiment. The core is equipped with ARM TrustZone technology, so we can implement virtualization using VMM. The above discussions show that RT685 is suitable for verifying the proposed method's essential part.

Figure 10 shows the circuit used for emulating the proposed method, and Fig. 11 shows its photo. The circuit is mainly constructed from four ICs and other components.

- LTC3542 is a buck converter to generate the supply voltage for the RTOS V_{RT}.
- AD5160 is a digital potentiometer inserted into the feedback path of the buck converter. We can control V_{RT} by adjusting the wiper of this potentiometer connected as a rheostat.
- ADG839 is an analog multiplexer to distribute V_{GP} and V_{RT} according to the signal from the MCU V_{sw}.

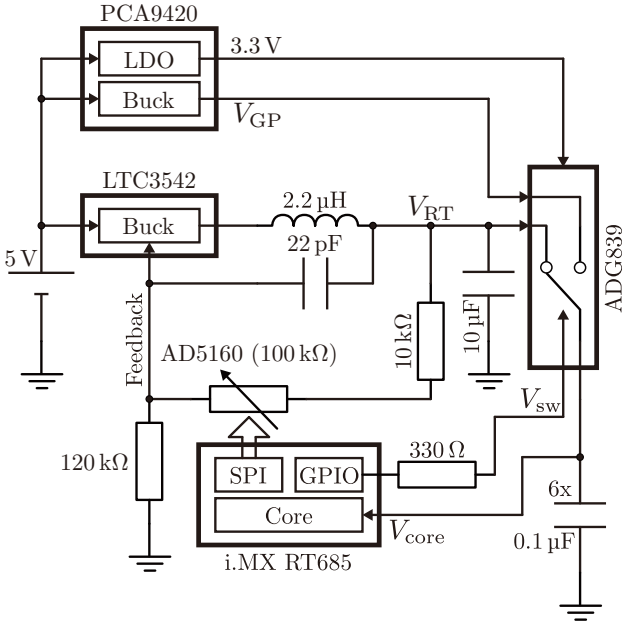


Fig. 10: The circuit for emulating DVFS virtualization (instantiation of Fig. 8).

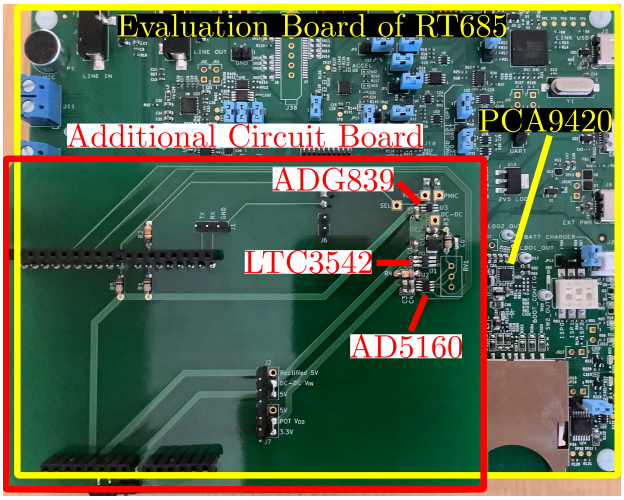


Fig. 11: Photo of the experiment cirtuit.

- PCA9420 is a designated power management IC for RT685, and its output is generally supplied to the core. In our experiment, we used the output of PCA9420 as the supply voltage for the GPOS V_{GP}.

B. Voltage transition latency

Firstly, we switched V_{core} from V_{GP} to V_{RT} while the CoreMark benchmark was running on the processor to verify that the circuit depicted in Fig. 10 works correctly. Figure 12 shows the voltage transition when V_{core} is switched from $V_{\mathrm{GP}} = 0.7\,\mathrm{V}$ to $V_{\mathrm{RT}} = 1.13\,\mathrm{V}$. The transition time when the control signal V_{sw} falls is treated as zero. Due to the load step of a DC-DC converter, V_{RT} and V_{GP} vary immediately, where V_{RT} rises and V_{GP} falls. V_{core} does not change rapidly because of the decoupling capacitors and takes roughly 500 ns to reach the level of V_{RT}. We also confirmed that the processor correctly operates while V_{core} is transitioning. General VMM

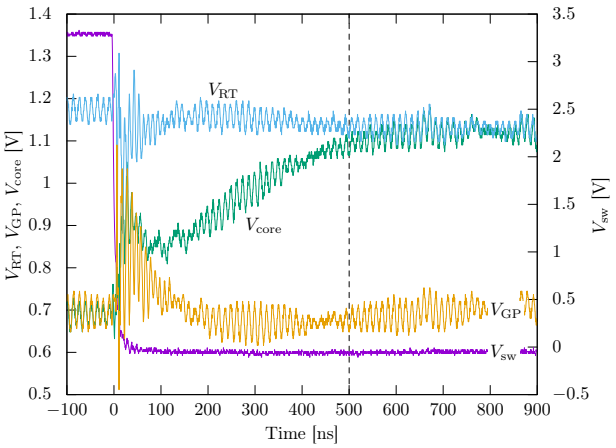


Fig. 12: Switching from 0.7 V, 75 MHz to 1.13 V, 300 MHz. The falling of the switching signal (V_{sw}) causes the change in the connection of the power supply from the GPOS (V_{GP}) to the RTOS (V_{RT}), and the supply voltage of the core (V_{core}) rises within 500 ns.

takes more than 1 µs to switch the context from the GPOS to the RTOS [2]–[6]. Therefore, the switching latency of V_{core} negligibly affects the real-time capability.

C. Emulation setup of proposed method

RT685 is not equipped with the hardware support discussed in Sect. IV. Therefore, we emulate the proposed method in the following manner.

Only one PLL can generate the core clock in RT685. Thus, we generate a fixed 300 MHz clock from the PLL and divide it by 1, 2, or 4 just before the core to realize immediate switching of 300 MHz, 150 MHz, and 75 MHz of the clock frequency. Although RT685 can operate at 275 MHz and 220 MHz, we do not use these operating conditions in this experiment since they cannot be generated from 300 MHz by an integer divider.

RT685 supports virtualization by ARM TrustZone technology; however, it cannot handle the supply voltage and clock. Therefore, we add a function on the software of the VMM to control the supply and clock upon distributing an interrupt to the RTOS and transitioning from the RTOS to the GPOS.

After V_{sw} is changed and the RTOS starts execution, the clock frequency must be increased when V_{core} is matched to V_{RT} completely. We controlled the clock as follows in this paper.

1) Start a timer when controlling V_{sw}.
2) A compare-match occurs after 500 ns are passed.
3) The DMA is triggered by the match and writes a divider setting to the memory-mapped register.

Note that when yielding to the GPOS, the supply and clock can be switched simultaneously.

D. Emulation results in energy reduction and overhead

This experiment verifies energy reduction and overhead introduced by adding the hardware. The CoreMark benchmark is executed on both the RTOS and GPOS, and we measure the energy consumption of the whole circuit. We applied the algorithm for the soft real-time GPOS, and it is run on the processor operated with a discrete set of frequencies, as discussed in Sect. III-A4. We selected CoreMark to evaluate the typical energy consumption of RT685 because NXP — the vendor of RT685— uses CoreMark for their energy consumption measurement.

In this experiment, we measure the energy consumption of all the components on the board, including the newly equipped DC-DC converter (LTC3542), the digital potentiometer (AD5160), and the analog multiplexer (ADG839), as well as existing essential parts (e.g., PCA9420, external flash, and others), to evaluate the energy overhead introduced by implementing the mechanism of immediately scaling the core voltage in addition to the energy consumption of the processor core. Note that both the experiments with and without the newly developed board employ the essential parts, and it consumes a considerable amount of power constantly regardless of the speed of the processor core.

Although the ultimate goal of this work is to minimize the energy consumption of the entire system on the board, this

experiment measures and evaluates the power consumption (energy per unit time as shown in the vertical axes in Fig. 14 and Fig. 15) instead of the energy consumption. We can convert the power consumption to energy consumption by dividing the measured power value by the frequency of the processor core. However, showing the energy consumption value obtained by this conversion may be misleading because the DVFS technique scales the operating voltage of the processor only and the components other than the processor are independent of the DVFS. Lower the core frequency, higher the energy consumption of the components other than the processor, which is misleading. Therefore, we show the power consumption (i.e., energy per unit time) on the vertical axes in Fig. 14 and Fig. 15. Note that the reduction rates of power consumption and energy consumption are equal.

1) Previous method: We measure energy consumption using the constant supply voltage of 1.15 V and clock frequency of 300 MHz with the unmodified evaluation board circuit depicted in Fig. 13. Figure 14 shows the change in energy consumption per unit time E_0/T for different CPU utilization rates of the RTOS U_{RT} and the GPOS U_{GP}. The energy consumption linearly and positively depends on U_{RT} and U_{GP} according to (1).

2) Proposed method: We measure energy consumption using the variable supply voltage of 0.7 V, 0.8 V, and 1.15 V

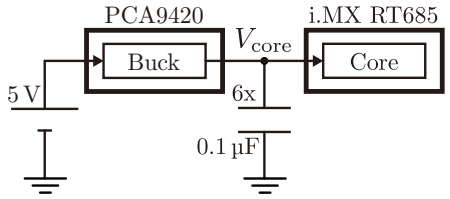


Fig. 13: The unmodified evaluation board circuit for the previous dual-OS platform.

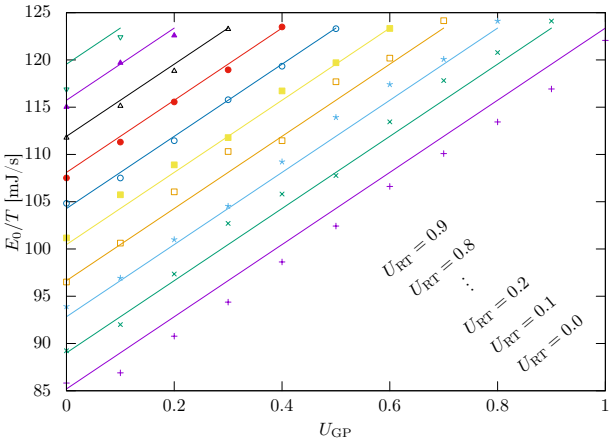


Fig. 14: Energy consumption per unit time of the conventional dual-OS platform for CoreMark benchmark, which corresponds to the result without applying DVFS. The energy consumption per unit time linearly and positively depends on U_{RT} and U_{GP} according to (1).

and clock frequency of 75 MHz, 150 MHz, and 300 MHz for the GPOS with the hardware of Fig. 10. We used the constant supply voltage of 1.13 V and clock frequency of 300 MHz for the RTOS as is deployed in the evaluation board of RT685.

Figure 15 shows the change in energy consumption E_{discrete}/T for different CPU utilization rates of the RTOS U_{RT} and the GPOS U_{GP}. E_{dvfs}/T is also plotted as dashed lines assuming that RT685 could operate with the continuously changing clock frequency. The actual energy consumption E_{discrete}/T is the straight lines connecting dots on the ideal curve of E_{dvfs}/T.

3) Energy reduction of proposed method: Figure 16 shows the change in energy reduction ratio for different CPU utilization of the RTOS U_{RT} and the GPOS U_{GP}. Smaller $U_{\mathrm{RT}}+U_{\mathrm{GP}}$ is, f_{GP} is likely to be slower, improving the energy reduction ratio. The maximum energy reduction is approximately 43 %. A notable observation from Fig. 16 is that the reduction ratio is always positive. This counterintuitive result claims that even if $V_{\mathrm{RT}} = V_{\mathrm{GP}} = 1.13\,\mathrm{V}$ (i.e., the maximum voltage), the energy consumption is reduced from the original hardware configuration depicted in Fig. 13, which uses 1.15 V as the supply voltage, although we added the hardware components.

Adding the hardware components of Fig. 10 introduces the following consumption.

- The static consumption of ADG839: According to the datasheet, the static current consumption of ADG839 is 3 nA. The supply voltage of 3.3 V results in static power consumption of 9.9 nW.
- The static consumption of AD5160: According to the datasheet, the static current consumption of AD5160 is 3 µA. The supply voltage of 3.3 V results in static power consumption of 9.9 µW.
- The loss of the on-resistance of ADG839: According to

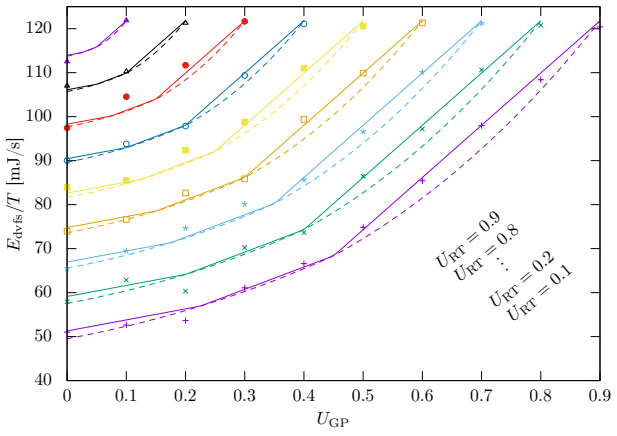


Fig. 15: Energy consumption per unit time of DVFS virtualization for CoreMark benchmark, which corresponds to the result of the soft real-time approach discussed in Sect. III-A. The ideal energy consumption plotted as dashed lines follow (2), and the actual energy consumption per unit time plotted as solid lines connect three points on the ideal curve by straight lines according to (4).

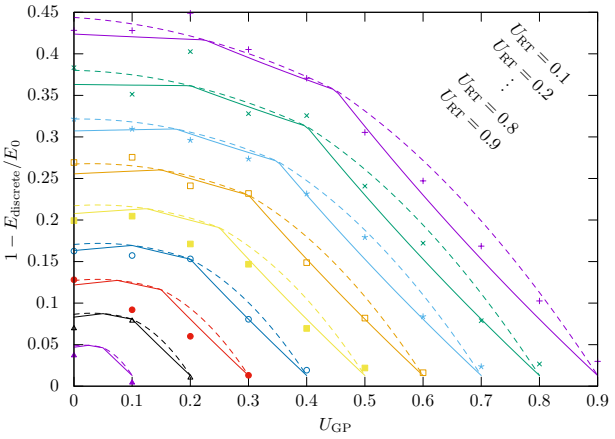


Fig. 16: Energy reduction of DVFS virtualization for CoreMark benchmark, which also corresponds to the results of the soft real-time approach discussed in Sect. III-A. DVFS virtualization reduces energy consumption for all conditions.

the datasheet, the on-resistance of ADG839 is typically $0.35\,\Omega$. Also, according to the datasheet of RT685, current consumption is roughly $50\,\text{mA}$ when executing CoreMark at $300\,\text{MHz}$. Therefore, the power consumption at the on-resistance is estimated at around $0.875\,\text{mW}$.

- The idling consumption of LTC3542: According to the datasheet, LTC3542 consumes $0.5\,\text{mA}$ when idle. The supply voltage of $5\,\text{V}$ results in idling power consumption of $2.5\,\text{mW}$.
- The idling consumption of PCA9420: According to the datasheet, PCA9420 consumes $0.7\,\mu\text{A}$ when idle. The supply voltage of $5\,\text{V}$ results in idling power consumption of $3.5\,\mu\text{W}$.

The sum of all the above power consumption values is only $3.4\,\text{mW}$, which is much smaller than the power consumption of the processor. Therefore, the energy overhead introduced by adding a power supply and switch is negligible.

The conversion efficiency of LTC3542 might be higher than PCA9420, and this causes energy reduction for the case $V_{\text{RT}} = V_{\text{GP}}$. Also note that although RT685 can operate at $300\,\text{MHz}$ with the supply voltage of $1.13\,\text{V}$, PCA9420 can only generate $1.15\,\text{V}$. On the other hand, the combination of LTC3542 and AD5160 can adjust the output voltage for about a $2\,\text{mV}$ step, realizing $300\,\text{MHz}$ operation of RT685 with the supply voltage of just $1.13\,\text{V}$.

The result of this experiment is summarized in the following.

- The power consumption overhead introduced by adding a power supply and a switch for DVFS virtualization is negligible.
- Applying DVFS virtualization to a commercial microprocessor gives the energy consumption expressed as (4). Automobile and avionics often comply with TDMA scheduling, which executes multiple software for statically allocated time windows [24]. ARINC-653 and AUTOSAR specify the typical TDMA scheduling scheme, which can easily measure accurate CPU utilization of the RTOS and GPOS at runtime, realizing to obtain similar results to those shown in Fig. 16 when applied.

E. Emulation results with industry applications

In this experiment, we applied the proposed method for several example applications provided by NXP. Like the experiment in Sect. V-D, the RTOS executes the CoreMark benchmark, but the GPOS executes one of the example applications. We selected the following four applications, representing the typical usage of RT685 that NXP expects.

- ML: An application that captures voice audio from the digital microphone on the board and detects words by a neural network.
- Crypto: An application that demonstrates the cryptographic functionality of the mbed-TLS library using a dedicated accelerator.
- GUI: An application that draws GUI animation on an external LCD using the LVGL library.
- Wi-Fi: An application that communicates with an external machine using a Wi-Fi module.

We applied the naive algorithm discussed in Sect. III-A1 to the above four applications because they do not have soft real-time constraints. Note that even this naive algorithm is not realizable without our proposed DVFS virtualization because existing dual-OS platforms take several tens of microseconds to change the speed of the processor, which may cause violating the deadlines of real-time tasks.

Figure 17 shows the change in the relative energy reduction of the proposed method for different CPU utilization rates of the RTOS U_{RT}. The smaller U_{RT} is, the more energy is reduced, and reduced energy decreases according to the increase of U_{RT}. Our method reduces energy consumption by $31\,\%$ on average. The maximum energy reduction is more than $60\,\%$, which is better than the result in Fig. 16. This experiment uses several peripherals and reduces their energy consumption, causing a higher energy reduction ratio than Fig. 16, which

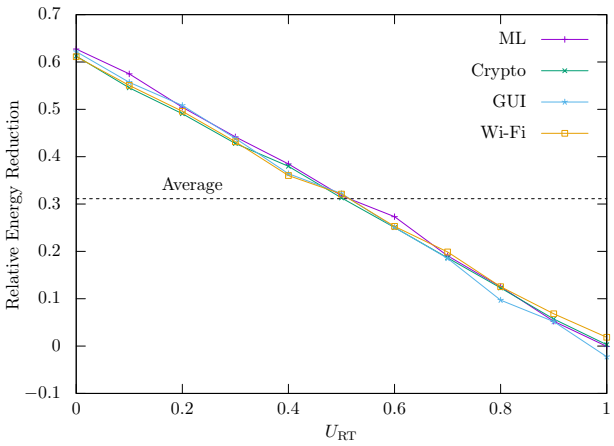


Fig. 17: Energy reduction of DVFS virtualization for several applications. DVFS virtualization reduces energy consumption by $31\,\%$ on average.

does not use a peripheral. Another notable observation from Fig 17 is that the reduction ratio of GUI application is below zero at $U_{\mathrm{RT}} = 1$. However, the increased energy is only $2\,\%$.

The summary of the experimental results is as follows;

- DVFS virtualization can be applied to various applications and does not prevent the correct operation.
- A high energy reduction ratio can be obtained for typical applications of RT685 regardless of their types.

VI. Conclusion

This paper proposed DVFS virtualization, which combines the DVFS algorithm to minimize the energy consumption of a dual-OS platform and the hardware structure to switch the performance immediately. The proposed DVFS algorithm efficiently reduces energy consumption by simultaneously running two different DVFS policies suitable for the RTOS and GPOS, respectively. The algorithm also automatically exploits the early completion of the RTOS than its WCET to save energy. In addition to the energy reduction, the proposed hardware realizes immediate switching of the supply voltage and clock frequency, which prevents the dual-OS from violating the real-time constraints of the RTOS even when the GPOS aggressively lowers the voltage to save energy. Also, our method can use general buck converters whose voltage conversion efficiency is typically more than $90\,\%$, although the voltage transition is slow. Such chopper DC-DC converters can efficiently produce fine-grained levels of voltages, better than integrated switched-capacitor DC-DC converters and voltage dithering, which are inefficient or can only produce discrete levels of voltages.

The evaluation experiment using the commercial processor board showed that the proposed hardware could change the supply voltage within $500\,\mathrm{ns}$ and reduce the energy consumption of typical applications by $60\,\%$ in the best case compared to traditional dual-OS platforms without preventing their correct operation. We also experimentally demonstrated that the energy consumption overhead introduced by the additional hardware is negligible, and the energy consumption is widely reduced following the theoretical formula.

Our future work includes developing a processor with hardware supports for the proposed DVFS virtualization, improving the time to change the performance, and applying the DVFS virtualization to multicore processors.

Acknowledgment

This work is supported by JST CREST Grant Number JPMJCR18K1.

References

[1] G. Heiser, "The Role of Virtualization in Embedded Systems," in *Workshop on Isolation and Integration in Embedded Systems*, 2008.

[2] Y. Kinebuchi, W. Kanda, Y. Yumura, K. Makijima, and T. Nakajima, "A Hardware Abstraction Layer for Integrating Real-Time and General-Purpose with Minimal Kernel Modification," in *Software Technologies for Future Dependable Distributed Systems*, 2009.

[3] B. Zuo, K. Chen, A. Liang, H. Guan, J. Zhang, R. Ma, and H. Yang, "Performance Tuning Towards a KVM-Based Low Latency Virtualization System," in *International Conference on Information Engineering and Computer Science (ICIECS)*, 2010.

[4] M. Liu, D. Liu, Y. Wang, M. Wang, and Z. Shao, "On Improving Real-Time Interrupt Latencies of Hybrid Operating Systems with Two-Level Hardware Interrupts," *IEEE Transactions on Computers*, 2011.

[5] D. Sangorrín, S. Honda, and H. Takada, "Reliable Device Sharing Mechanisms for Dual-OS Embedded Trusted Computing," in *International Conference on Trust and Trustworthy Computing (TRUST)*, 2012.

[6] P. Dong, A. Burns, Z. Jiang, and X. Liao, "TZDKS: A New TrustZone-Based Dual-Criticality System with Balanced Performance," in *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018.

[7] J. Yang, Y. Chen, H. Wang, and B. Wang, "A Linux kernel with fixed interrupt latency for embedded real-time system," in *International Conference on Embedded Software and Systems (ICESS)*, 2005.

[8] R. Rybaniec and P. Z. Wieczorek, "Measuring and minimizing interrupt latency in Linux-based embedded systems," in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments*, 2012.

[9] L. Perneel, H. Fayyad-Kazan, and M. Timmerman, "Can Android be used for real-time purposes?" in *International Conference on Computer Systems and Industrial Informatics (ICCSII)*, 2012.

[10] W. Kim, D. Brooks, and G.-Y. Wei, "A Fully-Integrated 3-Level DC-DC Converter for Nanosecond-Scale DVFS," *IEEE Journal of Solid-State Circuits*, 2012.

[11] B. Zimmer, Y. Lee, A. Puggelli, J. Kwak, R. Jevtić, B. Keller *et al.*, "A RISC-V Vector Processor With Simultaneous-Switching Switched-Capacitor DC–DC Converters in 28 nm FDSOI," *IEEE Journal of Solid-State Circuits*, 2016.

[12] F.-F. Ma, W.-Z. Chen, and J.-C. Wu, "A Monolithic Current-Mode Buck Converter With Advanced Control and Protection Circuits," *IEEE Transactions on Power Electronics*, 2007.

[13] N. Min Allah, Y.-J. Wang, J.-S. Xing, M. Nisar, and A.-R. Kazmi, "Towards Dynamic Voltage Scaling in Real-Time Systems-A Survey," *IJCSES International Journal of Computer Sciences and Engineering Systems*, 2007.

[14] S. Vestal, "Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance," in *International Real-Time Systems Symposium (RTSS)*, 2007.

[15] P. Huang, P. Kumar, G. Giannopoulou, and L. Thiele, "Energy efficient DVFS scheduling for mixed-criticality systems," in *International Conference on Embedded Software (EMSOFT)*, 2014.

[16] I. Ali, Y.-I. Jo, S. Lee, W. Y. Lee, and K. H. Kim, "Reducing Dynamic Power Consumption in Mixed-Critical Real-Time Systems," *Applied Sciences*, 2020.

[17] L. Behera and P. Bhaduri, "An Energy-Efficient Time-Triggered Scheduling Algorithm for Mixed-Criticality Systems," *Design Automation for Embedded Systems*, 2020.

[18] Y.-W. Zhang, "Energy-Aware Mixed-criticality Sporadic Task Scheduling Algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[19] B. Dominik, *CPUFreq Governors*. [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

[20] S. Akram, J. B. Sartor, and L. Eeckhout, "DVFS performance prediction for managed multithreaded applications," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2016.

[21] Y. Shakhsheer, S. Khanna, K. Craig, S. Arrabi, J. Lach, and B. H. Calhoun, "A 90nm data flow processor demonstrating fine grained DVS for energy efficient operation from 0.25V to 1.2V," in *Custom Integrated Circuits Conference (CICC)*, 2011.

[22] E. M. Benhani and L. Bossuet, "DVFS as a Security Failure of TrustZone-enabled Heterogeneous SoC," in *International Conference on Electronics, Circuits and Systems*, 2018.

[23] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "VoltJockey: Breaching TrustZone by Software-Controlled Voltage Manipulation over Multi-Core Frequencies," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[24] S. Saidi, S. Steinhorst, A. Hamann, D. Ziegenbein, and M. Wolf, "Special Session: Future Automotive Systems Design: Research Challenges and Opportunities," in *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2018.