JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 1121-1140 (2017)
DOI: 10.6688/JISE.2017.33.5.2

Optimal Energy Saving DVFS Approach of Embedded Processors*

WEN-YEW LIANG, MING-FENG CHANG AND YEN-LIN CHEN+
Department of Computer Science and Information Engineering
National Taipei University of Technology
Taipei, 10608 Taiwan
E-mail: ylchen@csie.ntut.edu.tw

Dynamic voltage and frequency scaling (DVFS) is considered one of the most efficient techniques for reducing energy consumption. Our studies have found that the optimal operation speed with optimal energy efficiency typically achieves when the processor running at a specific speed other than the lowest speed. Additionally, there exists an inverse relationship between the memory access rate (MAR) and the frequency that can minimize the energy consumption. Therefore, a predictive model can be deduced from the memory access rate to determine a frequency that tends to minimize the CPU energy consumption. In this paper, the existence of a critical speed and the memory access rate-critical speed equation (MAR-CSE) is proved theoretically and practically. A lower bound for Dynamic Voltage and Frequency Scaling is also defined. We report the most energy-efficient DVFS model that provide upper bounds of the reduction of energy consumption. The proposed approach has been implemented on embedded Linux operating system. The experimental results on energy consumption analysis show that our algorithm outperforms other existent DVFS algorithms.

Keywords: dynamic voltage and frequency scaling (DVFS), handheld devices, embedded systems, low power software design, optimal energy saving

1. INTRODUCTION

In nowadays, a lot of battery-powered handheld devices usually feature power managements for extended the using time. For example, because one of the major sources of power consumption is the LCD backlight, a device typically turns off the backlight when the display is not in use. Moreover, the CPU can be placed in an idle or sleep mode when no tasks need to be processed. These methods are called dynamic power management (DPM). One improvement to backlight power management involves the use of a sensor to control the brightness level according to the ambient light. Liang *et al.* [1] presented an adaptive workload driven power management policy with an objective to improve energy efficiency according to the corresponding workload variations. A similar method, called dynamic voltage and frequency scaling (DVFS) [2], can be applied to the processors when multiple voltage and frequency levels are supported. DVFS provides an energy-efficient mechanism while the CPU is in an active state, and is highly useful when a system does not need to run its applications at the highest performance point [3]. In this case, the frequency and voltage can be adjusted for reducing power consumption according to the computing system status and the task performance requirement. DVFS

Received May 7, 2015; revised June 21 and December 5, 2016; accepted January 9, 2017.
Communicated by Jan-Jan Wu.
+ Corresponding author.
* This work was supported by the Ministry of Science and Technology of Taiwan with Grant. No. MOST-105-2221-E-027-079.

also can be easily implemented in a real-time system in this way under the timing constraint, the task can be executed with a lower CPU frequency and voltage with satisfactory performance that can indeed reduce power dissipation [4, 5]. Moreover, the power managements [6, 7] that combine the DVFS and DPM approaches had been proposed for mobile devices.

This paper proposes an optimal energy saving DVFS approach and determines the upper bounds of the reduction of energy consumption on DVFS systems. Based on the experimental measurements, we have observed that reducing the CPU frequency may not always induce lower energy consumption. The lowest energy consumption usually occurs under a specific CPU frequency other than the lowest CPU frequency. This is due to the effect of memory accesses. The frequency that leads to the lowest energy consumption is achieved as the critical speed for the used program on the measured platform. The existence of this phenomenon could be mathematically proved in this study by observing the power consumption models of the processor and memory accesses. In addition, we also find an inverse relationship exists between the memory access rate (MAR) and the frequency of the minimized energy consumption as derived from the models and observed from the experiments. Thus, a correlation model based on the relationship can be constructed and used during the execution of tasks to predict the critical speed when the run-time MAR information is obtainable.

Based on the critical speed, a suitable CPU frequency that can balance the energy consumption and performance should be chosen as the target frequency. The system performance information for computing the memory access rate can be obtained from the hardware performance counters if available. The hardware performance counters are provided by typical processors, such as Intel XScale processors and ARM processors. A hardware counter is realized as the performance monitor unit (PMU) [8]. This paper implemented the proposed DVFS algorithm in the embedded platform with Linux operating system as a user-space power manager. This paper describes how to design and implement the algorithm.

The remainder of this paper is organized into seven sections described as follows. Related works are first discussed in Section 2. In Section 3, the existences of the critical speed and the relationship between the memory access rate and the critical speed are proved and derived, respectively. Section 4 presents the MAR and critical speed data from benchmark programs for constructing a data-based correlation equation. The proposed DVFS algorithm is also introduced and explained here. Section 5 describes the implementation of the algorithm as a Linux user-space power manager. Section 6 reports the experimental results and presents a comparison of the proposed algorithm with standard Linux DVFS implementations. Finally, the conclusion is provided in Section 7.

2. RELATED WORK

Most applications do not need to be run at the highest performance level from the processor at all times. Consequently, DVFS can be used to reduce the energy consumption of the running systems. The dynamic power caused by gate switching operations substantially contributes to the total power dissipated in the CMOS circuits. The dynamic power consumption can be expressed by

$$P_{dyn} = N_{sw} C_L V_{dd}^2 f \quad (1)$$

where N_{sw} is the switching activity, C_L is the load capacitance, V_{dd} is the supply voltage, and f is the operating frequency. Based on Eq. (1), the power consumption is proportional to the product of the frequency and the square of the supply voltage. Although decreasing the frequency can help reduce the power consumption, the program execution time is usually also lengthened accordingly. Therefore, the total energy consumption may remain the same. However, according to the relationship between the circuit delay and the voltage [9], a lower frequency typically allows a lower supply voltage needed to be used. Hence, decreasing the frequency and required voltage simultaneously could have a quadratic to cubic effect on the reduction of the power consumption, or equivalently, nearly a quadratic effect on energy reduction. This is the central idea behind DVFS mechanisms.

In applying DVFS, the processor may reduce its frequency along with the voltage when the performance demand is lower. For example, the Linux Ondemand DVFS algorithm [10] scales down the CPU frequency when the CPU utilization is decreased. DVFS is also often considered in real-time systems, in which the execution duration of a job can be extended to its maximal allowable finish time (*i.e.*, the deadline) so that the clock rate can be slowed, and the energy consumption can thus be reduced.

In real applications, executing the programs involves the operations not only inside the processor but also in memory accesses. DVFS researchers have begun examining the memory impact on the energy consumption of processors. Some researchers [11] indicated that the concept of memory accesses make great achievement by quantifying them into the DVFS algorithms. Lafond [12] proved that the distribution between the processor and memories is constant throughout the JVM execution with 70% of the energy consumed by memory accesses. This shows the importance of memory accesses on energy consumption determination, especially for embedded systems. In [13], Choi *et al.* presented a DVFS schema named workload decomposition, in which the system workload is decomposed into two parts: on-chip and off-chip. This technique is performed based on particular run-time statistics reported by the performance monitor unit counters. The on-chip workload represents the cycles of the executed instructions in CPU operations, and the off-chip workload represents the cycles for memory accesses. S. Eyerman [14] also used off-chip workload to propose a counter architecture for online DVFS profitability estimation on superscalar out-of-order processors. Isci [15] demonstrated a real-system implementation of a run-time phase predictor that works cooperatively with on-the-fly dynamic management, which considers the memory access rate as a critical factor. The paper in [16] proposed a continuously adaptive DVFS by using available performance counters to predict memory-bound rate with the effect of CPU speed.

Rajan *et al.* [17] used the MAR according to the run-time statistics information provided by the PMU to estimate the effect of external memory accesses. Chen [18] proposed dynamic frequency scaling schemes with approximation algorithms to provide tradeoffs of approximation guarantees in power consumption minimization with time and space complexity. The paper presented an online algorithm to achieve maximal energy savings by selecting the optimal CPU frequency-voltage combination based on the system workload. A dual-speed technique was adopted to approximate the optimal frequency-voltage combination by using two neighboring frequencies. Choi and Cha [19] also obtained number of data cache from PMU in XScale processor.

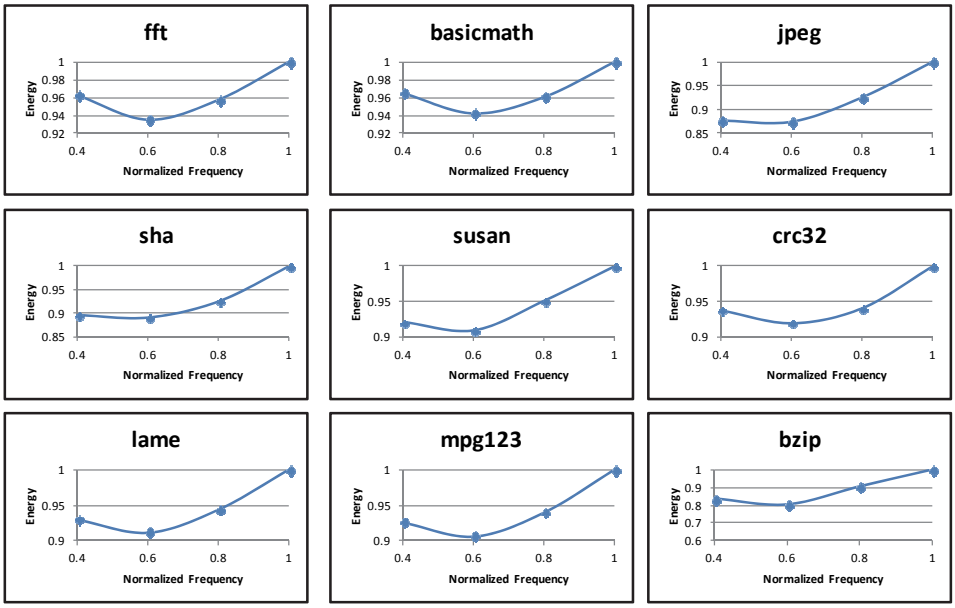


Fig. 1. Energy consumption for different frequencies. The CPU speed and the energy are normalized with respect to that of the highest frequency supported by the processor.

The reason that the frequency and voltage of a processor can be adjusted according to the effect of memory accesses is because memory operations usually pose a restriction to the execution time of a program due to memory access latency. Properly reducing the frequency of a running program that involves more memory accesses may only affect the performance slightly; in the meantime, considerable energy consumption could thus be saved. Similar ideas have been applied by using other power-saving methods. For example, the paper [20] proposes a performance evaluation method by using the memory access rate when DVFS is activated. In this way, users can easily achieve effective energy consumption and performance by specifying the allowable performance loss factor. Lin *et al.* [21] proposed a DVFS algorithm to reduce the power consumption through user feedback to maintain the performance at a satisfactory level.

Our work was motivated by the observations of real measurements on embedded devices. We observed that the lowest energy consumption usually appears at an operating frequency higher than the lowest frequency [22]. Several different features of benchmarks were run with different frequency on the PXA270 platform to observe the energy consumption. Fig. 1 shows an example of our measured result. As shown in the figure, as the CPU speed decreases from the highest frequency, the energy consumption is initially reduced. However, when the normalized speed is lower than a specifically speed, the energy consumption increases instead. This is because when the CPU slows down, the total execution time is accordingly lengthened and may hence cause the total energy consumption to be increased because the memory is still kept active and consumes extra energy during the extended execution time. Based on this example, the CPU speed at the turning point of the curve in Fig. 1 is called the critical speed of the program on the measured platform. To prove that the critical speed exists, a mathematical model for the

relationship between the processor speed and energy consumption is derived in the following section.

This paper presents a mathematical model to prove that the lowest energy consumption usually appears at an operating speed higher than the lowest frequencies. This study proposes an efficient memory-aware DVFS algorithm based on a relationship between the memory access behavior, which is determined according to the MAR, and the critical speed. An equation, called the memory access rate-critical speed equation (MAR-CSE), is constructed according to the relationship and used in run-time to predict the critical speed. The predicted frequency may somewhat decrease the execution performance because the equation is used to minimize the energy consumption rather than to maintain the level of performance. The main contributions of our work are listed as follows: (1) we prove the existence of the critical speed theoretically and experimentally; (2) we prove that the MAR is inversely proportional to the critical speed, and construct an equation based on this relationship, the MAR-CSE; (3) a lower boundary for Dynamic Voltage and Frequency Scaling is defined; (4) we also propose an optimal energy-saving DVFS algorithm for non-real-time systems. The algorithm was implemented as a user-space policy manager on the Linux for dynamic processor power management, in which the PMU is used to obtain run-time information on memory accesses for critical speed prediction. The algorithm and the implementation are detailed described in the following sections.

To develop an efficient method for applying an energy-saving algorithm to processors, power measurement is also a crucial issue because the accuracy of the prediction equation depends on the results of the measurements for the target processor. In [23], Rethinagiri proposed proposes a simulation based power estimation and optimization tool for complex processor based platforms at system-level (PETS). However, the tool still has a maximum prediction error of 4.5%. Contreras and Martonosi [24] demonstrated a linear power estimation model that uses the PMU to estimate the power consumption of an Intel PXA255 processor. The model provides a simple way to obtain power consumption information without requiring specific hardware to obtain the measurements. In [25], Snowdon *et al.* built a performance and energy model in which the PMU was also used as a mechanism to estimate the performance degradation and energy consumption. The power measurement becomes more important than ever on computing systems. By comparison, the goal of this study is to accurately measure the power consumption data for designing an accurate prediction mode, we used standalone measurement hardware instead of the estimation methods proposed by the previous works. The details of our improved measurement method are discussed in the subsequent section.

3. MODELS OF MAR AND CS

This section presents the existences of the critical speed and the relationship between the memory access rate and the critical speed.

3.1 The CPU Energy and the Frequency

Memory Access Rate [16] was used in the experiments ass an appropriate indication of the system workload. The MAR is defined as follows:

$$MAR = \frac{N_{mem}}{N_{instr}} = \frac{\text{Instruction Cache Missed + Data Cache Misses}}{\text{Number of Instructions Executed}} \tag{2}$$

$$N_{mem} = MAR \cdot N_{instr} \tag{3}$$

In these equations, N_{mem} is the number of memory accesses and N_{instr} is the number of instruction executed, assuming the bus frequency and the power consumption of the memory subsystem are both fixed. In real hardware, the CPU frequency is typically a multiple of the memory frequency:

$$f_{cpu} = m \cdot f_{mem} \tag{4}$$

In the equation, the coefficient m can be used to represent the CPU frequency because the memory frequency is assumed to be fixed. Let E_{cpu} be the energy consumption of the CPU, P_{cpu} the power consumption of the CPU, and T_{total} the total execution time. Based on Eq. (1), the following equations can be inferred from E_{cpu}.

$$E_{cpu}=P_{cpu} \cdot T_{total} = k \cdot C \cdot V^2 \cdot f_{cpu} \cdot T_{total} \tag{5}$$

The total execution time of a program T_{total} is written as:

$$T_{total} = T_{cpu} \cdot T_{mem} = \frac{N_{instr}}{f_{cpu}} + \frac{N_{mem}}{f_{mem}}. \tag{6}$$

We assume that T_{cpu} and T_{mem} cannot overlap. Therefore, Eq. (6) can be brought to Eq. (5):

$$E_{cpu} = k \cdot C \cdot V^2 \cdot f_{cpu} \cdot \left(\frac{N_{instr}}{f_{cpu}} + \frac{N_{mem}}{f_{mem}} \right) = k \cdot C \cdot V^2 \cdot f_{cpu} \cdot \left(N_{instr} + m \cdot N_{mem} \right). \tag{7}$$

As the processor frequency is reduced, the supply voltage can also be reduced. Therefore, the processor frequency is proportionate to the supply voltage ($f \propto v$). In Eqs. (7) to (8a), we substitute the squared voltage for $\lambda \cdot f_{cpu}^2$.

$$E_{cpu}=k \cdot C \cdot \lambda \cdot f_{cpu}^2 \cdot \left(N_{instr} + m \cdot N_{mem} \right) \tag{8a}$$

$$=k \cdot C \cdot \lambda \cdot f_{mem}^2 \cdot \left(N_{instr} \cdot m^2 + N_{mem} \cdot m^3 \right) \tag{8b}$$

$$= k \cdot C \cdot \lambda \cdot f_{mem}^2 \cdot N_{instr} \cdot \left(m^2 + MAR \cdot m^3 \right) \tag{8c}$$

Eq. (8) can be replaced with

$$E_{cpu}=\alpha \cdot \left(m^2 + MAR \cdot m^3 \right) \tag{9}$$

where $\alpha = k \cdot C \cdot \lambda \cdot f_{mem}^2 \cdot N_{instr}$. According to this derivation, the CPU energy has a cubic relationship with m. As the frequency increases, the energy consumed by the CPU

grows drastically cubically.

3.2 The Memory Energy and the Frequency

For the memory energy, the memory energy consumption can be derived as follows.

$$E_{mem}=P_{mem}\cdot T_{total}=P_{mem}\cdot\left(\frac{N_{instr}}{f_{cpu}}+\frac{N_{mem}}{f_{mem}}\right) \tag{10a}$$

$$=P_{mem}\cdot\left[\frac{N_{instr}}{(m\cdot f_{mem})}+\frac{N_{mem}}{f_{mem}}\right] \tag{10b}$$

$$=\frac{P_{mem}}{f_{mem}}\cdot N_{instr}\cdot\left(\frac{1}{m}+MAR\right) \tag{10c}$$

In the equation, E_{mem} and P_{mem} each represent the power consumption of memory. Assume $\beta=\frac{P_{mem}}{f_{mem}}\cdot N_{instr}$,

$$E_{mem}=\beta\cdot\left(\frac{1}{m}+MAR\right). \tag{11}$$

In other words, the energy consumption for the memory is inversely proportional to the frequency factor m. When the CPU speed is decreased, the execution time increases, resulting in the increase of memory energy consumption.

3.3 Modeling for Finding the Minimum Energy

Third, the energy consumption from Eqs. (9) and (11) can be combined to obtain the total energy consumption.

$$E_{total}=E_{cpu}+E_{mem}=\alpha\cdot\left(m^2+MAR\cdot m^3\right)+\beta\cdot\left(\frac{1}{m}+MAR\right) \tag{12}$$

Based on the equation, the total energy consumption is a function of the *MAR* and m. According to the definition of α and β in the previous two sections, we may treat them as constants, given a particular hardware platform. Based on fixing the value of the *MAR*, the relationship between m and E_{total} is depicted as Fig. 2, which also shows a plot of Eq. (12) in a fixed *MAR*.

Fig. 2 illustrates the relationship between the frequency and the energy consumption which has already been shown in Fig. 1 from the experiment. This relationship proves that the lowest energy consumption may not occur at the lowest frequency.

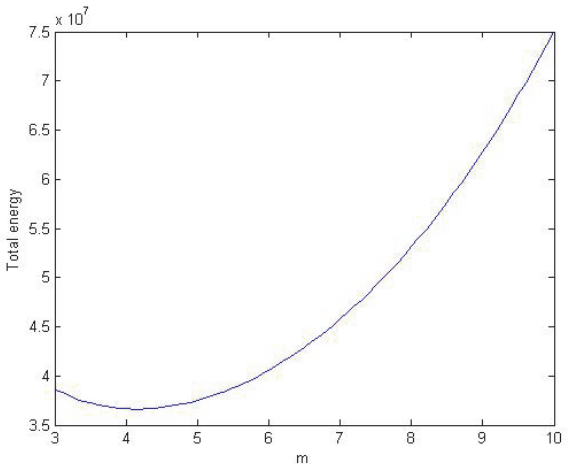


Fig. 2. The relationship between m (representing the CPU frequency) and the total energy consumption.

3.4 MAR and the Critical Speed

Let CS (critical speed) be the frequency (*i.e.*, the minimum value m here) such that $E_{total}(MAR, m)$ is minimized. The critical speed can be derived by finding the solution to $\min\{E_{total}(MAR, m), \forall m\}$, as shown in the following differential equations.

$$E_{total} = \alpha \cdot \left(m^2 + MAR \cdot m^3\right) + \beta \cdot \left(\frac{1}{m} + MAR\right) \tag{13a}$$

$$\frac{\partial E_{total}}{\partial m} = \alpha \cdot \left(2 \cdot m + 3 \cdot MAR \cdot m^2\right) + \beta \cdot \left(\frac{1}{m^2}\right) = 0 \tag{13b}$$

$$MAR = \frac{\alpha/\beta - 2 \cdot m^3}{3 \cdot m^4} \tag{13c}$$

From Eq. (13), we may further conclude that the MAR is inversely proportional to the critical speed. In other words, ($MAR \propto C/m$), *e.g.*, if the task is a memory-bound job, then the critical speed would be relatively low. By contrast, if it is a CPU-bound job, then the critical speed would be relatively high. Fig. 3 illustrates the relationship between the critical speed and the MAR (Eq. (13c)). The figure can be divided into three areas: above the curve (pointless DVFS area), below the curve (Meaningful DVFS area) and on the curve line (optimal energy line). If the DVFS scheme selects the CPU speed in the pointless DVFS area then the energy and performance would increased. However, picking the CPU speed in the meaningful DVFS area would get the better result that having the performance consideration. The optimal energy consumption falls on the critical speed line that called the optimal energy line. The optimal energy line also defines the lower boundary for the target frequency during the DVFS process.

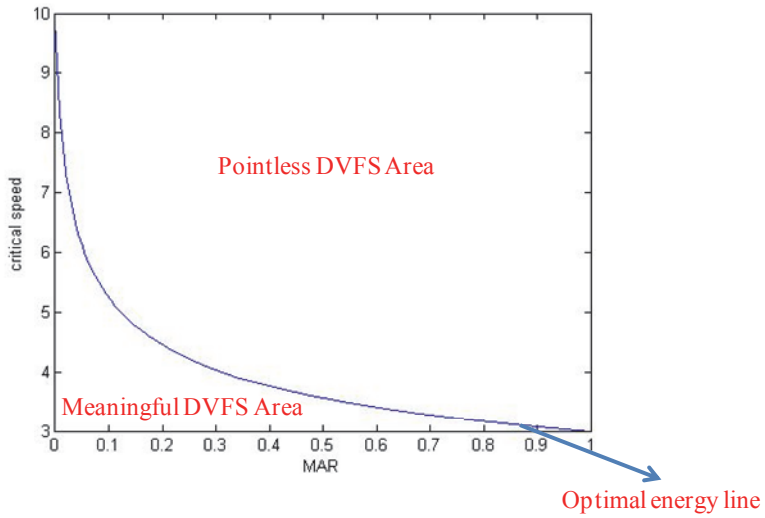


Fig. 3. The curve of m the critical speed and the MAR.

4. THE OPTIMAL ENERGY SAVING DVFS ALGORITHM

In the previous section, the MAR is proved to be inversely proportional to the critical speed. To model the relationship between the MAR and the critical speed accurately for the hardware platform that we used, experiments were arranged to collect related data. As described in the above sections, the run-time performance information for the MAR can be obtained from the hardware performance counters. A higher MAR value indicates that the task is a memory-bound program, whereas a lower value is for a CPU-bound program. For a memory-bound program, the external memory access time may dominate the total execution time. In such case, the CPU voltage and frequency could be slowed without significantly affecting the performance.

Table 1 lists the measured critical speeds and the MAR values of several benchmarks in the experiment. The critical speed is normalized with respect to the maximum CPU frequency of the platform. The MiBench [26] was chosen as the benchmark suite. The benchmarks include basicmath, bitcount, fft, sha, susan, jpeg, and mad. Considering that the problem size may affect how a program behaves, each benchmark program was tested with a larger and smaller amount of input data. The table in Fig. 4 clearly shows that the MAR is inversely proportional to the critical speed. A benchmark program with a higher MAR (which tends to be memory-bound) has a lower critical speed, while a program with a lower MAR (which tends to be CPU-bound) has a higher critical speed. It indicates that a program with a higher MAR may have a greater potential to consume fewer energy at a lower frequency. However, the energy consumed by a CPU-bound program may increase if the CPU frequency is further decreased, because it has a higher critical speed.

Table 1. Critical speed and MAR for the benchmarks. Program names appended with _b and _s are for big and small data size, respectively. The environment platform is Creator PXA270 board with the Linux kernel 2.6.25.

Benchmarks	MAR	Critical Speed
bitcount_b	0.000171	0.785
fft_b	0.001489	0.7628
bitcount_s	0.00182	0.7377
fft_s	0.00236	0.754
basicmath_s	0.002597	0.7201
sha_b	0.003209	0.6685
mad_b	0.003319	0.6567
susan_b	0.003708	0.6421
basicmath_b	0.005748	0.6888
mad_s	0.00848	0.5885
susan_s	0.009569	0.5806
sha_s	0.010571	0.5305
jpeg_b	0.013905	0.5854
jpeg_s	0.015367	0.56

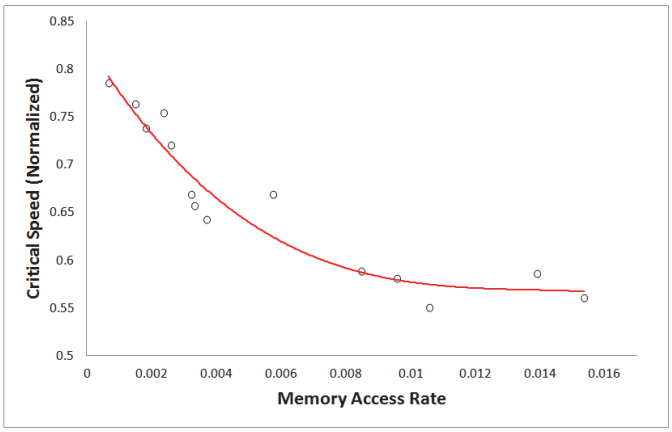


Fig. 4. Curve of the MAR-CSE equation.

To formally describe the correlation between the MAR and the critical speed, we used an approximation equation using the least square curve fitting method based on the data we collected. The equation is the MAR-CSE, which can be used to predict the criti-

cal speed during program execution time while the run-time MAR information of a program can be obtained. Fig. 3 depicts the approximation curve. Once the MAR value has been obtained at run time from the PMU hardware counters, the ideal frequency can be calculated according to the curve of the MAR-CSE. Because the prediction equation is a continuous function, the computed ideal frequency must first be mapped onto a discrete frequency supported by the processor. One way to solve this issue is to use the dual speed approach, which approximates the ideal frequency by using two neighboring discrete frequencies [17].

The concept of the proposed DVFS algorithm is to determine the working frequency from the MAR-CSE for achieving the goal of lowest energy consumption. The computational procedures of the proposed algorithm are depicted in Table 2. The proposed algorithm is an on-line dynamic voltage and frequency scaling algorithm, which accounts for the MARs. When the DVFS process is enabled, the algorithm collects statistical data from the PMU and calculates the MAR (at line 5). Because the overhead caused by the proposed MAR-CSE algorithm may possibly influence the CPU performance, a pre-computed lookup table is used instead in the computation (at line 6). From the lookup table, the frequency region in which the critical speed exists can be obtained immediately as the MAR value being obtained. The critical speed is then determined through the interpolation computation. After that, the proposed DVFS scheme used the dual speed method to approximate the optimal frequency-voltage combination by using two real neighboring frequencies that approximates the ideal operational frequency due to the voltage and frequency setting are not continuous in the real platform (at line 7).

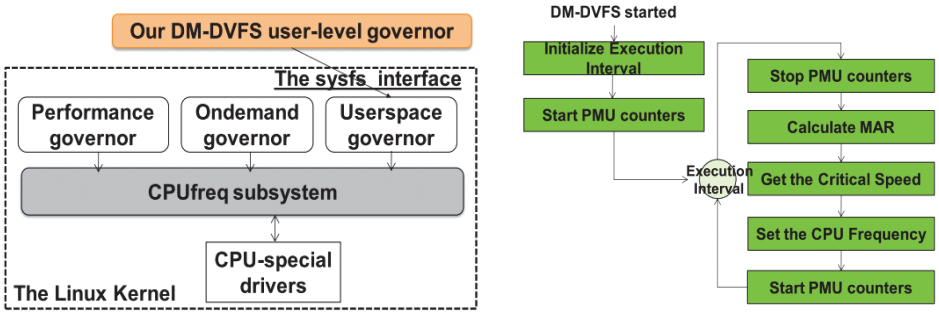
Table 2. The proposed DVFS algorithm.

	DVFS algorithm
Requirement: The counter information in PMU	
1.	initialize PMU counter registers.
2.	pmu_start() for first times calculate.
3.	for every execution interval do
4.	pmu_stop();
5.	calculate MAR (cache misses/instruction count)
6.	cs = get_critical_speed_by_lookup_table(MAR);
7.	target_freq = map_to_target_freq(cs);
8.	set_freq(target_freq);
9.	pmu_start();
10.	end for

5. IMPLEMENTATION

This paper implemented the proposed DVFS algorithm on the Linux kernel 2.6.25. The Linux kernel provides a modularized interface to manage the CPU frequency in the CPUfreq subsystem. The power management policy based on DVFS is called a “Gover-

nor" in Linux, which controls the frequency through the CPUfreq interface. CPUfreq decouples the driver of the CPU-specific hardware from the policies. There are several DVFS governors have been supported by Linux for power management. In the DVFS governors, the Performance governor scales the CPU at the highest frequency, and the Ondemand governor manages the CPU frequency based on the CPU utilization. The Userspace governor exports the available CPU frequency information to the user space and allows the user-level algorithm to control the CPU frequency through the Linux sysfs interface. The proposed DVFS algorithm called the DM-DVFS governor that was implemented as a Linux user-space daemon. The DM-DVFS calculates the current MAR value and chooses a proper frequency based on the MAR-CSE. The algorithm gets the PMU hardware counters and sets the CPU frequency through the Linux sysfs interface, under which we implemented the kernel-level supporting code. Fig. 5 (a) illustrates the structure of the Linux governors and our governor.



(a) CPU freq subsystem. (b) DM-DVFS governor.

Fig. 5. (a) Structure of the Linux CPUfreq subsystem and our governor; (b) Execution flow of the DM-DVFS.

Fig. 5 (b) shows how the DM-DVFS governor works. Once the DM-DVFS daemon is started, it periodically retrieves the cache-miss and instruction count information from the PMU through the sysfs interface to calculate a MAR value, then applies the MAR-CSE to determine the desired ideal frequency (*i.e.* the critical speed), with respect to the MAR value. The determined ideal frequency is then adoped as the target CPU frequency to be applied to the frequency (and the voltage) control hardware, also through the sysfs interface. The length of the DVFS algorithm execution interval affects the performance and power consumption according to the different platforms due to the different idle power consumption. This paper experimented different interval for the PXA270 platform on 50ms, 100ms and 200ms. The activation interval of this periodical DVFS management task is set to 200ms.

DVFS technology may bring further overhead for the system in energy consumption [9]. The MAR-CSE method is an equation with floating-point operations; its calculation might costs some additional computing resources, especially in an embedded system with limited computing resources, as well as the concern of extra energy consumption overhead. To effectively reduce the computational overhead, the proposed MAR-CSE equation is implemented and approximated as a pre-built lookup table, so that an

approximated solution can be rapidly determined by the interpolation method. The partitioning granularity of the curve for the table lookup may affect the approximated features. If the partitioning granularity is too much so the computational overhead will increased and also course the energy consumption increased. This paper experimented different of granularity for the MAR to find the reliable lookup table to reduce the computational overhead without too much effect to the energy consumption. Fig. 6 shows that the MAR-CSE curve is divided into 16 levels for ease of computation, and in the meantime to reduce the size of the lookup table. For example, if the MAR is located at level 8 (0.008 to 0.009), then the lookup table will pick the averaged critical speed in the corresponding level (0.6).

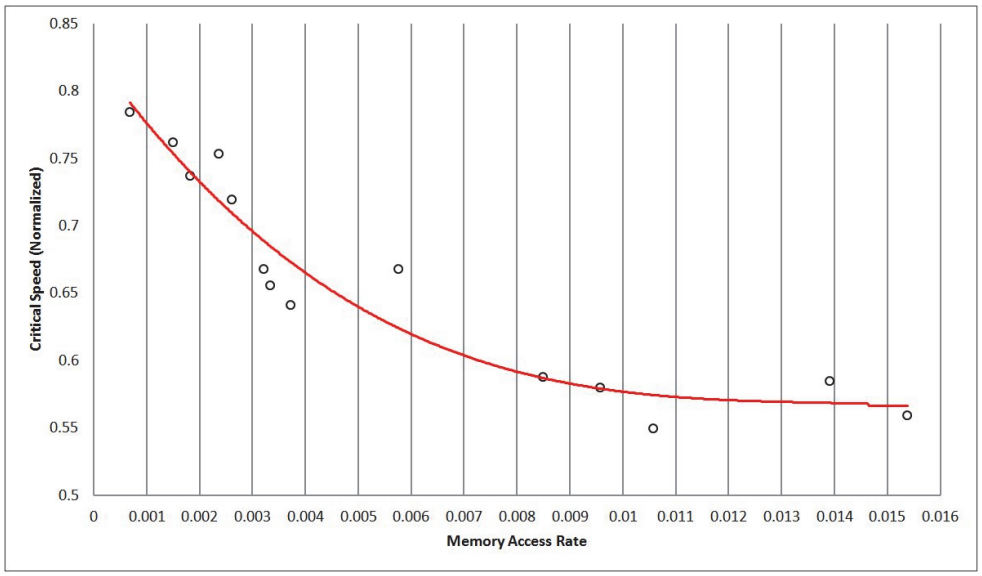


Fig. 6. Partitioning of the curve for table lookup.

6. EXPERIMENTAL RESULTS

This section introduces our experimental environment and presents the performance evaluation results on various benchmark programs and real applications of the proposed DVFS method.

6.1 The Experiment Environment

The experiments were implemented on an embedded platform, the Creator PXA270 development board with the Linux kernel 2.6.25. The PMIC board was used to support dynamic voltage scaling for the PXA270 development board. If the frequency is changed, then the corresponding voltage is altered accordingly. A high-performance data acquisition instrument (NI USB-6251) was used to collect the power data at a rate of 1000 samples per second. The voltage and the current of the CPU and the SDRAM were measured

to compute their power consumption. Fig. 7 illustrates the configuration of the measurement environment.

The MAR and the critical speed data that were used to create the MAR-CSE were measured on this platform. To conduct comparative experiments, four DVFS governors were measured, including the Linux Ondemand governor, the Linux Performance governor, the Powersave governor, the Conservative governor, and our DM-DVFS governor. This paper defines the critical speed as the frequency at which the energy consumption can be minimized. When a frequency lower than the critical speed is used, then the energy consumption becomes less efficient and the performance might also worsen. As mentioned in the previous section, the critical speed is used as a lower boundary for the dynamic voltage and frequency scaling operation. The following subsection reports our results from the benchmark programs and two other real applications.

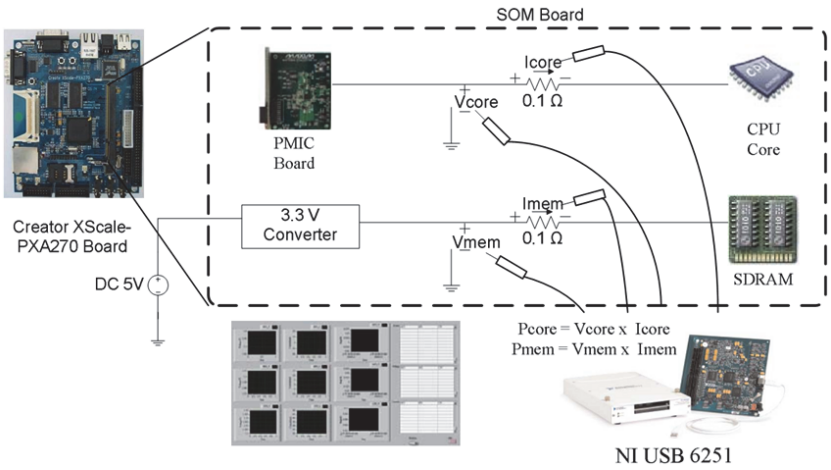


Fig. 7. Measurements of the Creator PXA270 components.

6.2 Performance Evaluation Results

An important issue of the DVFS algorithm is computational complexity. A DVFS implementation in the operation systems has two primary overheads: algorithmic execution intervals and DVFS computational overheads. To measure the overheads introduced by the proposed DM-DVFS algorithm, a Linux kernel information, jiffies, is used. Fig. 8 reports the jiffies before and after the DM-DVFS algorithm being performed. We can observe that the overhead is tiny during benchmark processes. The overhead analysis results are shown in Fig. 8. As can be seen from the analytical results, the basicmath_b only has 0.25% overhead within whole executing time. The worst case in our experiments is around 0.87%. By comparison, the overhead of the Linux Ondemand governor are 0.55% to 1.45%. As a result, the presented DM-DVFS algorithm is relatively lightweight and can provide high computational efficiency.

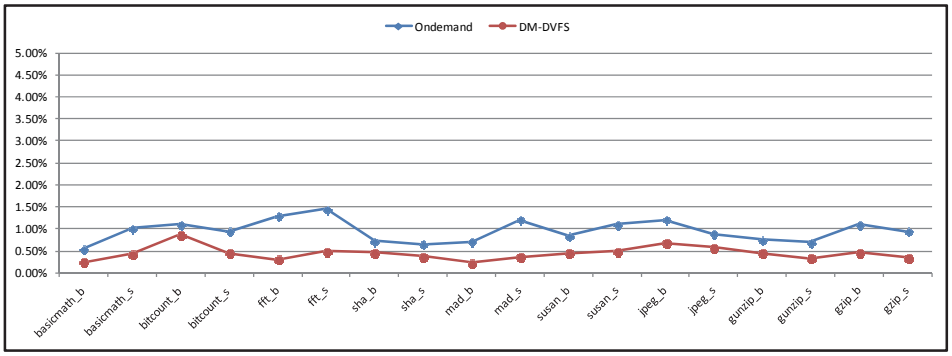


Fig. 8. Overhead of DM-DVFS algorithm.

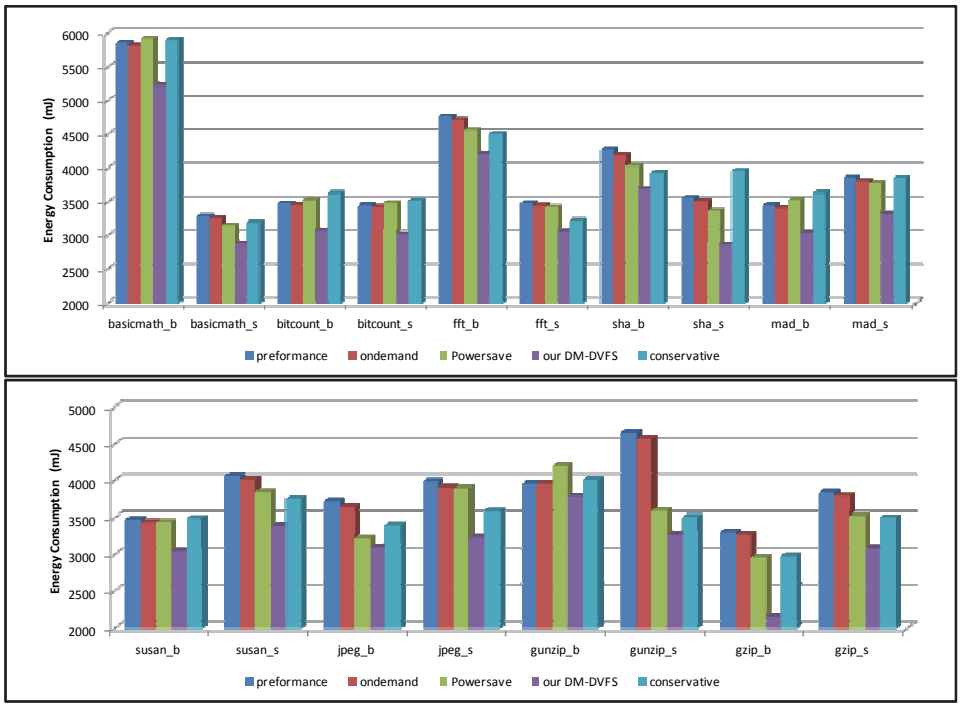


Fig. 9. Energy consumption of the benchmarks.

Fig. 9 shows the total energy consumption of the benchmark programs. Based on the result, the energy consumption of the Ondemand governor is still close to that of the Performance governor. On average, it saves only 1.2% of energy consumption as compared to the Performance governor. The Conservative governor saves 1.1% to 15% of energy consumption. By contrast, the proposed DM-DVFS technique significantly reduces energy consumption by 9% to 42%. Our DM-DVFS shows the best results on energy consumption because the DM-DVFS adopts the critical speed as a lower boundary of the target frequency. When the DVFS governor uses a frequency less than the critical

point, then the energy usage becomes as inefficient as the lowest frequency governor. The Powersave governor not only consumed higher energy than the proposed DM-DVFS governor but also obtained the worst energy consumption results on the CPU-bound benchmarks because of the extremely long execution times.

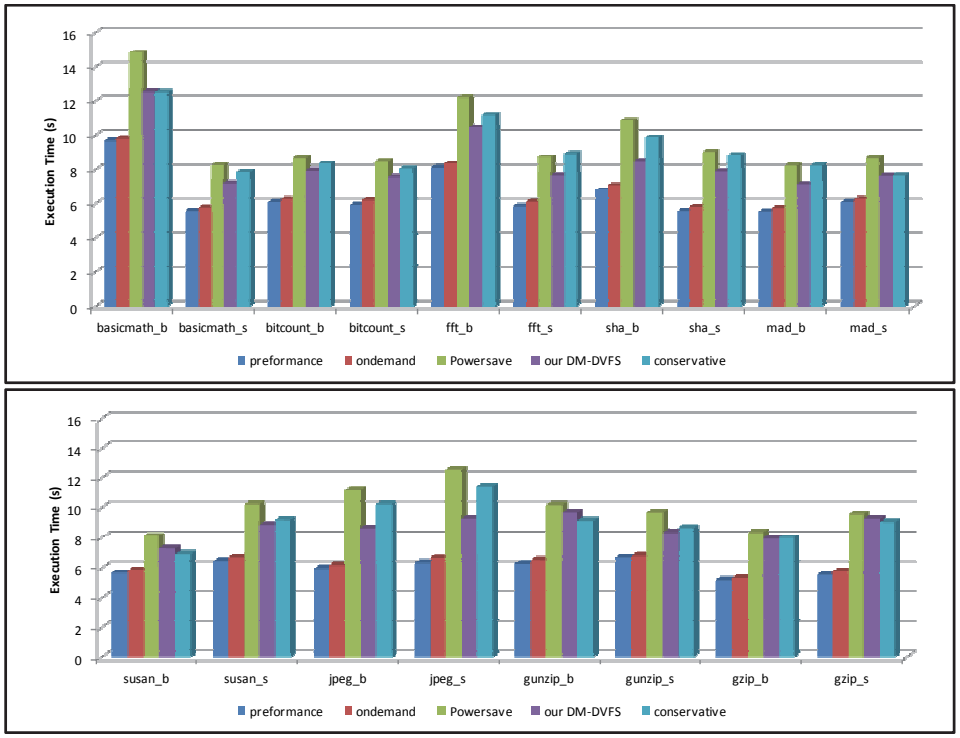


Fig. 10. Execution time of the benchmarks.

The execution time of the benchmark programs is shown in Fig. 10. The Performance governor has the best performance because it always runs at the highest frequency. The Ondemand governor runs slightly longer than the Performance governor does because the frequency is adjusted according to the CPU utilization. For the proposed DM-DVFS governor, the execution time for the DM-DVFS is much longer than that for the Ondemand governor (by 29% on average). This is because the DM-DVFS governor is targeted at minimizing the energy consumption instead of maintaining the performance. On the other hand, the Powersave governor has a longer execution time because it always fixes at the lowest frequency. As mentioned previously, when the frequency is lower than the critical speed, the energy consumption may increase. Consequently, the Powersave governor induced the highest energy consumption among all the governors in CPU-bound benchmarks.

The improvements of the energy consumption and the execution time usually appear in different aspects. A lot of researches use the energy-delay production (EDP) as a fair comparison metric. However, this study focused on optimizing and minimizing energy consumption. Thus, EDP and execution time are not suitable for comparative advertising.

Further experimental results on real applications were also investigated, including bzip2 and PictureLoader. They are file compression and photo browsing applications, respectively. Fig. 11 shows the results of their energy consumption and execution time. From the results, we can see that the proposed DM-DVFS shows the most favorable results in energy consumption because DM-DVFS uses the critical speed as a lower boundary of the target frequency. Regarding the execution time, the values for DM-DVFS are between the other governors. These results indicate that the proposed DM-DVFS governor can provide more efficient results than those of other governors when energy consumption is considered the first priority.

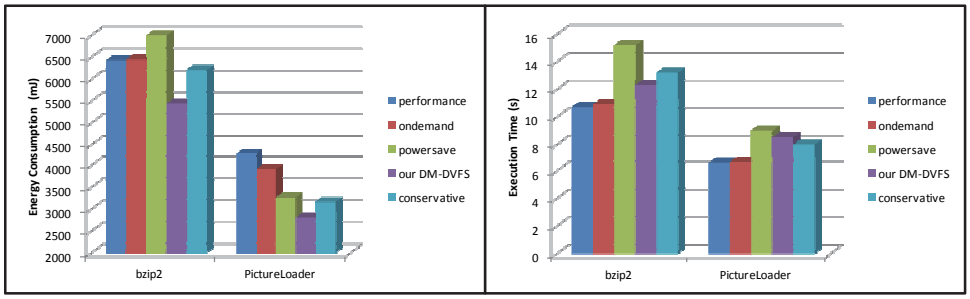


Fig. 11. Energy consumption and execution time of bzip2 and PictureLoader.

7. CONCLUSION

This paper proves the existence of a critical speed for minimal energy consumption. In addition, a relationship between the memory access rate and the critical speed was empirically derived as a prediction formula, called the "MAR-CSE" model. In considering that a frequency lower than the critical speed may increase the energy consumption inversely, we proposed an optimal energy saving DVFS algorithm that predicts the critical speed and uses it as the lower boundary for the target frequency during the DVFS process.

To predict an optimal CPU speed to achieve the best energy consumption efficiency, the MAR-CSE was used to analyze the relationship between MAR and the critical speed. During the task execution period, a dynamic MAR value was obtained from the PMU hardware counters of the embedded platform and then applied to the equation to find the appropriate frequency that tended to minimize the energy consumption. The DM-DVFS was realized on a PXA270 embedded platform, on which we ported the Linux kernel 2.6.25 as a Linux user-space governor for the purpose of DVFS management.

The experimental results demonstrate that our DM-DVFS governor can effectively save the lowest energy consumption by 9% at best to 42% at worst for the studied benchmark programs. The execution time for the DM-DVFS is longer than that for the Ondemand governor because the DM-DVFS governor is used to minimize the energy consumption but not to maintain the performance. From the performance evaluation on two real applications, the obtained results of energy savings also validate the efficiency and feasibility of the proposed method. In our future research works, we will involve the features of the input and output behaviors of the application programs in the DVFS algo-

rithm, to provide better energy saving performance on various applications and newly developed embedded and handheld platforms.

REFERENCES

1. A. Liang, L. Xiao, and L. Ruan, "Adaptive workload driven dynamic power management for high performance computing clusters," *Computers and Electrical Engineering*, Vol. 39, 2013, pp. 2357-2368.
2. N. K. Jha, "Low-power system scheduling, synthesis and displays," *IET Computers and Digital Techniques*, Vol. 152, 2005, pp. 344-352.
3. J. Choi and H. Cha, "System-level power management for system-on-a-chip-based mobile devices," *IET Computers and Digital Techniques*, Vol. 4, 2010, pp. 400-409.
4. C. C. Yang, K. Wang, M. H. Lin, and P. Lin, "Energy efficient intra-task dynamic voltage scaling for realistic CPUs of mobile devices," *Journal of Information Science and Engineering*, Vol. 25, 2009, pp. 251-272.
5. J. Wu and K. L. Ke, "Energy efficient intra-task dynamic voltage scaling for realistic CPUs of mobile devices," *Journal of Information Science and Engineering*, 2014, pp. 765-786.
6. S. O. Park, J. K. Lee, J. H. Park, and S. J. Kim, "Adaptive power management system for mobile multimedia device," *IET Communications*, Vol. 6, 2012, pp. 1407-1415.
7. G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Transactions on Embedded Computing Systems*, Vol. 13, 2014, pp. 1-21.
8. Intel, *Intel XScale Core Developer's Manual*, Technical specification, 2004, doc No. 273473-002, http://download.intel.com/design/intelxscale/27347302.pdf, 2014.
9. T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, Vol. 25, 1990, pp. 584-594.
10. V. Palladi and A. Starikovskiy, "The ondemand governor: past, present and future," in *Proceedings of Linux Symposium*, 2006, pp. 223-238.
11. S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of energy-cognizant scheduling techniques," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, 2013, pp. 1447-1464.
12. S. Lafond, "Energy consumption analysis for two embedded Java virtual machines," *Journal of System Architecture*, Vol. 53, 2007, pp. 328-337.
13. K. Choi, R. Soma, and M. Pedarm, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade-off based on the ratio of off-chip access to on-chip computation times," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, 2005, pp. 18-28.
14. S. Eyerman and L. Eeckhout, "A counter architecture for online DVFS profitability estimation," *IEEE Transactions on Computers*, Vol. 59, 2010, pp. 1576-1583.
15. C. Isci, G. Contreras, and M. Martonosi, "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchi-*

tecture, 2006, pp. 359-370.
16. V. Spiliopoulos, S. Kaxiras, and G. Keramidas, "Green governors: A framework for continuously adaptive DVFS," in *Proceedings of International Green Computing Conference and Workshops*, 2011, pp. 1-8.
17. D. Rajan, R. Zuck, and C. Poellabauer, "Workload-aware dual-speed dynamic voltage scaling," in *Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2006, pp. 251-256.
18. J. J. Chen, K. Huang, and L. Thiele, "Dynamic frequency scaling schemes for heterogeneous clusters under quality of service requirements," *Journal of Information Science and Engineering*, Vol. 28, 2012, pp. 1073-1090.
19. J. Choi and H. Cha, "System-level power management for system-on-a-chip-based mobile devices," *IET Computers and Digital Techniques*, Vol. 4, 2010, pp. 400-409.
20. W.-Y. Liang, M.-F. Chang, Y.-L. Chen, and J.-H. Wang, "Performance evaluation for dynamic voltage and frequency scaling using runtime performance counters," *Applied Mechanics and Materials*, Vols. 284-287, 2013, pp. 2575-2579.
21. B. Lin, A. Mallik, P. Dinda, G. Memik, and R. Dick, "User- and process-driven dynamic voltage and frequency scaling," in *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software*, 2009, pp. 11-22.
22. W.-Y. Liang, Y.-L. Chen, and M.-F. Chang, "A memory-aware energy saving algorithm with performance consideration for battery-enabled embedded systems," in *Proceedings of IEEE International Symposium on Consumer Electronics*, 2011, pp. 547-551.
23. S.-K. Rethinagiri, O. Palomar, O. Unsal, A. Cristal, R. Ben-Atitallah, and S. Niar, "PETS: Power and energy estimation tool at system-level," in *Proceedings of the 15th International Symposium on Quality Electronic Design*, 2014, pp. 535-542.
24. G. Contreras and M. Martonosi, "Power prediction for Intel XScale® processors using performance monitoring unit events," in *Proceedings of International Symposium on Low Power Electronics and Design*, 2005, pp. 221-226.
25. D. C. Snowdon, S. M. Petters, and G. Heiser, "Accurate on-line prediction of processor and memory energy usage under voltage scaling," in *Proceedings of the 7th ACM & IEEE International Conference on Embedded Software*, 2007, pp. 84-93.
26. M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free commercially representative embedded benchmark suite," in *Proceedings of IEEE International Workshop on Workload Characterization*, 2001, pp. 3-14.

Wen-Yew Liang (梁文耀) received his BS degree in Computer Science and Engineering from Tatung Institute of Technology, Taiwan, in 1992, his MS degree in Computer Science from National Tsing Hua University, Taiwan, in 1994, and his PhD degree in Computer Science and Information Engineering from National Taiwan University in 1998. From 1998 to 2000, he served as a communication officer in the Department of Defense, Taiwan. He worked for Avant! Corporation as an EDA software engineer from 2000 to 2001. Between 2001 and 2004, he joined an embed-

ded system design company, WISCORE Inc., and experienced the positions of a R&D staff engineer, the department manager, and the vice president. From 2005 to 2012, he transferred to National Taipei University of Technology as an Assistant Professor in the Department of Computer Science and Information Engineering. Currently, Dr. Liang is the founder and the technical director of AndroLead Co., Ltd., Taiwan. He was a member of ACM, IEICE, and IICM. His research interests include embedded systems, operating system design, low power system design, parallel and distributed systems, mobile computing, and parallelization for scientific applications.

Ming-Feng Chang (張銘峰) received the Ph.D. degree in Computer Science and Information Engineering from National Taipei University of Technology, Taipei, Taiwan, in 2016. His research interests include embedded systems, low power system design and computer performance evaluation.

Yen-Lin Chen (陳彥霖) received the B.S. and Ph.D. degree in Electrical and Control Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2000 and 2006, respectively. From February 2007 to July 2009, he was an Assistant Professor at the Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan. From August 2009 to July 2015, he was an Assistant Professor and Associate Professor at the Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan, and since August 2015, he is now a full Professor in the same institute. Dr. Chen is a Senior Member of the IEEE, and a member of IAPR, and IEICE. In 2012, he earned the best annual paper award sponsored by the Intelligent Transportation Society of Taiwan. In 2003, he received Dragon Golden Paper Award sponsored by the Acer Foundation and the Silver Award of Technology Innovation Competition sponsored by the AdvanTech. He also earned the Silver Invention Award from Ministry of Economic Affairs of Taiwan, and received best annual ITS paper awards two times in 2011 and 2012. His research interests include image and video processing, embedded and wearable systems, intelligent video analytics, intelligent vehicles, and intelligent transportation system. His research results have been published on over 90 journal and conference papers.