韓國컴퓨터情報學會 論文誌
第16卷 第6號, 2011. 6.

2011-16-6-1-2

실시간 비디오 처리에 적합한 에너지 효율적인 멀티코어 스케쥴링

백 형 구*, 여 정 모*, 이 완 연**

Energy-Efficient Multi- Core Scheduling for Real-Time Video Processing

Hyung Goo Paek*, Jeong Mo Yeo*, Wan Yeon Lee**

요 약

본 논문에서는 DVFS 기능을 제공하는 멀티코어 프로세서 상에서 실시간 비디오 태스크의 에너지 소모량을 최소화하는 최적 스케쥴링 기법을 제안한다. 제안된 스케쥴링 기법은 멀티코어의 병렬처리 기법을 활용하도록 적절한 수의 멀티코어들을 태스크의 수행에 할당하고, 사용되지 않는 코어들의 전원을 끄며, 실시간 태스크의 데드라인을 만족하는 최저 클락 주파수를 배정한다. 단일 코어에서 태스크를 실행하는 기존 방법과 그리고 모든 코어들에서 태스크를 실행하는 기존 방법을 제안된 스케쥴링 기법과 비교하는 실험 결과에서, 제안된 스케쥴링 기법이 기존 방법들의 에너지 소모량을 각각 최대 67%, 89% 감소시킴을 확인하였다.

▸ Keyword : 저전력 설계, 멀티코어 프로세서, 태스크 스케쥴링, 실시간 시스템

Abstract

In this paper, we propose an optimal scheduling scheme that minimizes the energy consumption of a real-time video task on the multi-core platform supporting dynamic voltage and frequency scaling. Exploiting parallel execution on multiple cores for less energy consumption, the propose scheme allocates an appropriate number of cores to the task execution, turns off the power of unused cores, and assigns the lowest clock frequency meeting the deadline. Our experiments show that the proposed scheme saves a significant amount of energy, up to 67% and 89% of energy consumed by two previous methods that execute the task on a single core and on all cores respectively.

▸ Keyword : low-power design, multi-core processor, task scheduling, real-time system

• 제1저자 : 백형구, 교신저자: 이완연
• 투고일 : 2010. 12. 24, 심사일 : 2011. 01. 21, 게재확정일 : 2011. 02. 13
* 부경대학교 컴퓨터공학과 (Dept. of Computer Engineering, Pukyong National University)
** 동덕여자대학교 컴퓨터학과 (Dept. of Computer Science, Dongduk Women's University)
※ This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2009-0064347).

I Introduction

"Video on mobile"[1,16] is becoming the norm for on-demand TV. The limited battery life on mobile devices becomes a burning issue, as demand for video playing time increases. Therefore, an energy-efficient solution specifically tailored to a long-lived video task running on mobile devices must be explored. In devising an appropriate solution, an emerging trend in processor design that draws out attention is the multi-core platform. A chip containing tens of processing cores is commercialized and viable processing cores in a chip will increase rapidly with advances in microprocessor design [2].

Another processor design trend, which coalesces with the multi-core platform to improve energy efficiency, is advanced energy management with dynamic voltage and frequency scaling (DVFS) [3,4,17]. The speed of the DVFS-enabled processor is linearly proportional to the clock frequency, whereas the energy consumption increases proportionally to a polynomial function of the clock frequency. The degree of this polynomial function is typically not less than two, even though the exact relationship between the energy consumption rate and the clock frequency depends on the hardware [3-5]. Consequently, putting more cores into a real-time task, while lowering the clock frequency, can open a rich possibility of reducing the total energy consumption of the task.

In this paper, we consider the problem of energy-efficient video processing based on the aforementioned trends in processor design. While the multi-core platform has been intensively investigated for energy-efficient processing of many real-time tasks, it has been rarely considered for a single real-time task due to its overabundant hardware. This paper addresses how to exploit these overabundant resources with the DVFS capability. Specifically, we propose an energy-efficient off-line scheduling designed for a periodic long-lived video task on the DVFS-enabled multi-core platform; the video task represents the most popular and energy-demanding application for current and future mobile devices. The application of the proposal is not limited to the video tasks, but may be applicable to other long-lived periodic real-time tasks.

The main contribution of this paper is to introduce an optimal off-line scheduling that minimizes energy consumption of a periodic real-time task by fully exploiting the architectural benefits of multiple cores and DVFS capability. Considering the non-linearly scaling property of parallel execution and the irregular energy consumptions of discrete frequencies, the proposed scheduling determines both the number of cores allocated to parallel execution and the frequency executing each computation cycle under the deadline constraint. This is analogous to the 3-D bin-packing problem to determine a triple input: number of cores, frequency value and execution time, whereas previous scheduling studies [3-5,7-9,10,11] are restricted to the 2-D bin-packing problem to determine a duplex input: number of processors and execution time, or frequency value and execution time. The proposed scheduling allocates a pertinent number of cores to task execution, turns off the power of the other unused cores, and assigns the lowest frequency meeting the deadline. If the lowest frequency is not one of the available discrete frequencies, it is generated with two adjacent discrete frequencies. Our evaluation shows that the proposed scheduling saves significant energy, up to 67% and 89% of the energy consumed by two greedy approaches: executing the task on a single core while turning off the other cores, and executing the task in parallel on all available cores, respectively.

Numerous previous studies [3-5,7-10] have investigated the energy minimization of real-time tasks on multiple processing elements (PLs). However, they missed the rich energy-saving capability of parallel execution exploiting o verabundant PLs. Yang et al. [11] addressed an

assignment problem of interdependent subtasks to heterogeneous processors working on different but fixed frequencies (speeds), whose combination is equivalent to a virtually single DVFS-enabled processor. Only a few recent studies [12,13] considered the parallel execution on multiple DVFS-enabled PLs. Li et al. [12] addressed an on-line adaptation that dynamically changes the activated cores executing subtasks of an application and the frequency supplied to the cores. Our previous work [13] addressed a greedy scheduling that distributes all available cores evenly to independent concurrent tasks. However, these methods do not achieve the minimal energy consumption and include impractical assumptions or severe runtime overhead: infinitely continuous frequencies with their enforced energy consumption formulas [13], frequent power on/off of cores during idle time [12], and parallel execution on changeable cores [12,13]. In contrast, this paper addresses an optimal scheduling that achieves the minimal energy consumption of a real-time parallel task under practical restrictions discretely available frequencies with their arbitrary energy consumptions, one-time power off of unused cores, and parallel execution on fixed cores.

II PRELIMINARIES

1. Task Model

A real-time video task consists of consecutive image frames arriving from the network or being retrieved from the disk. Because image frames require different computation cycles, task scheduler accounts for the worst case, i.e., the maximum computation cycles, for simplicity. The computation cycles of each frame must be completed within a given time limit, i.e., deadline D. The deadline depends on the type of media and coding. For instance, NTSC DVD quality MPEG-2 video can be transmitted at approximately 30 or 24 frames per second, for such cases the deadline is given by $D \approx 33.3ms$ and $D \approx 41.7ms$, respectively.

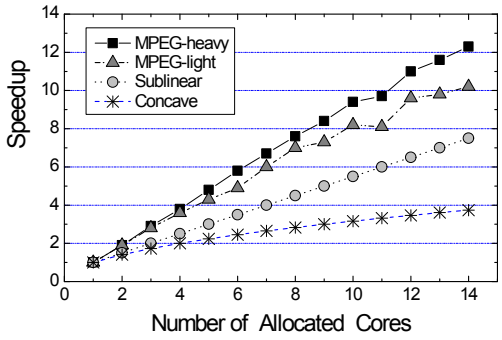

Fig. 1. Four Speedup vectors of parallel processing

Video decoding tasks can be partitioned into multiple computational components, e.g., separate groups of image pictures and disjoint partitions of each image picture [6]. These components can be computed concurrently and independently on multiple processing cores. The speedup of parallel execution is approximately proportional to the number of allocated cores, but usually less than the number of allocated cores due to inefficiency factors of parallel execution, such as intrinsically sequential processing portion, conflict of concurrent memory accesses, unbalanced load of subtasks, and additional communication between subtasks. To examine the impact of various speedups of parallel execution, we use four speedup models depicted in Fig. 1. The first two speedup models are drawn from the experimental data generated in the parallel MPEG-2 video task on the Silicon Graphics Challenge multiprocessor [6]. They are 1408 × 960 and 352 × 240 resolution video tasks, and we call them MPEG-heavy and MPEG-light, respectively. We synthetically generate the other two that represent the tasks including severe inefficiency factors of parallel execution; The task, whose inefficiency factors occupy half the total computation, is called Sublinear. The speedup of

Sublinear task with n allocated cores is $S_{sb}[n] = (n-1) \times 0.5 + 1$ for $n \geq 1$. The task, whose portion of the inefficiency factors to the total computation increases along the number of allocated cores, is called Concave. The speedup of Concave task with n allocated cores is $S_{cn}[n] = \sqrt{n}$ for $n \geq 1$.

2. Processor Model

An identical frequency is supplied to all activated cores in the processor. The unused cores are powered off to save energy [9,10]. Core processing speed is proportional to the supplied clock frequency. In terms of execution time, the completion time is the required computation cycles divided by the clock frequency. The K discrete frequencies available to these processors are denoted as $f_1, ..., f_K$ in increasing order. For a given frequency f_k where $1 \leq k \leq K$, the power consumption is denoted as p_k. Then, the energy consumption and execution time of each cycle are $\frac{p_k}{f_k}$ and $\frac{1}{f_k}$, respectively. If $i < j$, then $f_i < f_j$ and $p_i < p_j$. Even when a processor has no computation to execute, the power consumption in the idle status, i.e., leakage power consumption, is strictly positive [5,10]. The power consumption in the idle status is denoted as p_0 For convenience, we additionally define a virtual frequency of the idle status as f_0 and set its value to zero, because its computation speed is semantically equivalent to zero.

Table 1. Processor Model

k	0	1	2	3	4	5
f_k (MHz)	0	150	400	600	800	1000
Voltage (V)	0.75	0.75	1.0	1.3	1.6	1.8
p_k (mW)	40	80	170	400	900	1600

For the practical DVFS evaluation, we use the data obtained from a well-known DVFS processor, the Intel XScale [5]. Table I shows the available frequencies, their voltages and power consumptions (energy consumption rates) of the processor for the instruction execution.

3. Problem Formulation

The problem tackled in this paper is to minimize the total power consumption of a real-time task on N homogeneous cores. If n cores are assigned to the parallel execution, the other $(N-n)$ cores are powered off to save energy. The task requires at most C cycles and should be completed within the deadline D. Speedup values $S[n]$ of parallel executions on n cores for $1 \leq n \leq N$ are given in advance. When C cycles are executed in parallel on n cores with a speedup of $S[n]$, the task can be executed within at most $\lceil \frac{C}{S[n]} \rceil$ cycles. If the completion time of all $\lceil \frac{C}{S[n]} \rceil$ cycles under the minimum frequency f_1 is earlier than the deadline D, the power p_0 of the idle status is consumed for the slack time $(D - \lceil \frac{C}{S[n]} \rceil \cdot \frac{1}{f_1})$.

We do not turn off the power of the activated cores during the slack time, because the time delay and extra energy required to turn off/on the power [14] are relatively large, compared with the tight deadline of image frames. In addition, we do not change the number of cores allocated to a task during execution, because it incurs severe overhead such as subtask preemption, migrations of suspended subtasks and synchronization. A schedule is referred to as n-feasible if it allocates n cores to the task execution and completes the task execution within the deadline D. An n-feasible schedule is called n-Optimal Schedule if it consumes the minimum energy amongst all n -feasible schedules.

III Proposed Scheduling Scheme

First, the proposed scheduling scheme excludes the defective frequency f_y that violates the convex function

of power consumption. That is, $\frac{p_y - p_x}{f_y - f_x} > \frac{p_z - p_y}{f_z - f_y}$ for any $f_x < f_y < f_z$. Based on Lemma 1, the defective frequency is discarded henceforth. Calculation results of $\frac{p_k - p_{k-1}}{f_k - f_{k-1}} > \frac{p_{k+1} - p_k}{f_{k+1} - f_k}$ for $1 < k <$ K can select all defective frequencies. Note there is no defective frequency in Table I.

Lemma 1: If $\frac{p_y - p_x}{f_y - f_x} > \frac{p_z - p_y}{f_z - f_y}$ for $f_x < f_y < f_z$, the frequency f_y is not used in any n-Optimal Schedule.

Proof: Let us assume that any n-Optimal Schedule uses the frequency f_y to execute C_y cycles. If $\frac{C_a + C_b}{f_y} = \frac{C_a}{f_x} + \frac{C_b}{f_z}$ where $C_a + C_b = C_y$, then $C_b \cdot \frac{f_x}{f_x - f_y} = C_a \cdot \frac{f_z}{f_y - f_z}$. If $\frac{p_y - p_x}{f_y - f_x} > \frac{p_z - p_y}{f_z - f_y}$ and $C_b \cdot \frac{f_x}{f_x - f_y} = C_a \cdot \frac{f_z}{f_y - f_z}$, then $C_a \cdot (\frac{p_x}{f_x} - \frac{p_y}{f_y}) - C_b \cdot (\frac{p_y}{f_y} - \frac{p_z}{f_z}) = (\frac{p_x}{f_x} \cdot C_a + \frac{p_z}{f_z} \cdot C_b) - \frac{p_y}{f_y} \cdot (C_a + C_b) < 0$. That is, the energy consumption using f_y to execute $(C_a + C_b)$ cycles, $\frac{p_y}{f_y} \cdot (C_a + C_b)$, is larger than that using f_x to execute C_a cycles and f_z to execute C_b cycles, $(\frac{p_x}{f_x} \cdot C_a + \frac{p_z}{f_z} \cdot C_b)$. Consequently, the frequency f_y is not used in any n-Optimal Schedule. ▣

Next, the proposed scheme calculates the energy consumptions of all n-Optimal Schedules for $1 \le n \le N$, and selects the best schedule with the least energy consumption from the n-Optimal Schedule. Each n-Optimal Schedule chooses a frequency f_m that is nearest to and no smaller than $\left\lceil \frac{C}{S[n]} \right\rceil \cdot \frac{1}{D}$, from $f_1, ..., f_K$. If $\left\lceil \frac{C}{S[n]} \right\rceil \cdot \frac{1}{f_m} = D$, it is clear that the schedule executing the task on n cores with the frequency f_m completes the C cycles within the time D and consumes the minimum energy among n-feasible schedules. In case $\left\lceil \frac{C}{S[n]} \right\rceil \cdot \frac{1}{f_m} < D$, the slack time $(D - \left\lceil \frac{C}{S[n]} \right\rceil \cdot \frac{1}{f_m})$ can be utilized to save more energy using a combination of another frequency f_{m-1}. Namely, the n-Optimal Schedule assigns two different frequencies f_m and f_{m-1}, where f_m is the smallest available frequency satisfying $f_m \ge \left\lceil \frac{C}{S[n]} \right\rceil \cdot \frac{1}{D}$. It attempts to find the transition point C' that satisfies $\frac{C'}{f_m} + \frac{\left\lceil \frac{C}{S[n]} \right\rceil - C'}{f_{m-1}} = D$. The transition point C' can be found as follows:

$$C' = \frac{f_m \cdot (\left\lceil \frac{C}{S[n]} \right\rceil - D \cdot f_{m-1})}{f_m - f_{m-1}} . \quad (1)$$

Because $\frac{\lceil C' \rceil}{f_m} + \frac{\left\lceil \frac{C}{S[n]} \right\rceil - \lceil C' \rceil}{f_{m-1}} \le D$ when $f_m > f_{m-1}$, it assigns the frequency f_m to execute $\lceil C' \rceil$ cycles and the frequency f_{m-1} to execute the remaining $(\left\lceil \frac{C}{S[n]} \right\rceil - \lceil C' \rceil)$ cycles. The following theorem shows that the n-Optimal Schedule assigns f_m to $\lceil C' \rceil$ cycles and f_{m-1} to $(\left\lceil \frac{C}{S[n]} \right\rceil - \lceil C' \rceil)$ cycles.

Theorem 1: The n-Optimal Schedule assigns f_m to $\lceil C' \rceil$ cycles and f_{m-1} to $(\left\lceil \frac{C}{S[n]} \right\rceil - \lceil C' \rceil)$ cycles.

Proof: We refer to the schedule, which assigns f_m to $\lceil C' \rceil$ cycles and f_{m-1} to $(\left\lceil \frac{C}{S[n]} \right\rceil - \lceil C' \rceil)$ cycles, as Original Schedule. Let us assume that there is another n-Optimal Schedule other than the Original Schedule. Then, the assumed n-Optimal Schedule assigns other frequencies, except f_m and f_{m-1}, to execute some cycles from the C cycles. If the frequency f_{m1}, such that $f_{m1} > f_m$, substitutes for f_m or f_{m-1}, then the

assumed n-Optimal Schedule consumes more energy than the Original Schedule. If the frequency f_{m2}, such that $f_{m2} < f_{m-1}$, substitutes for f_m or f_{m-1}, then the assumed n-Optimal Schedule cannot meet the deadline. Finally consider the case where f_{m1} and f_{m2} are used to execute some C_a cycles from the C' cycles and C_b cycles from the $(\lceil \frac{C}{S[n]} \rceil - \lceil C' \rceil)$ cycles, respectively. When $\frac{C_a}{f_{m1}} + \frac{C_b}{f_{m2}} = \frac{C_a}{f_m} + \frac{C_b}{f_{m-1}}$, the assumed n-Optimal Schedule consumes the minimum energy. If $\frac{C_a}{f_{m1}} + \frac{C_b}{f_{m2}} = \frac{C_a}{f_m} + \frac{C_b}{f_{m-1}}$ and $\frac{p_{m-1} - p_{m2}}{f_{m-1} - f_{m2}} \le \frac{p_m - p_{m-1}}{f_m - f_{m-1}} \le \frac{p_{m1} - p_m}{f_{m1} - f_m}$, then $(\frac{p_{m1}}{f_{m1}} \cdot C_a + \frac{p_{m2}}{f_{m2}} \cdot C_b) \ge (\frac{p_m}{f_m} \cdot C_a + \frac{p_{m-1}}{f_{m-1}} \cdot C_b)$. Then the energy consumption of the assumed n-Optimal Schedule, $(\frac{p_{m1}}{f_{m1}} \cdot C_a + \frac{p_{m2}}{f_{m2}} \cdot C_b)$, is no smaller than that of the Original Schedule, $(\frac{p_m}{f_m} \cdot C_a + \frac{p_{m-1}}{f_{m-1}} \cdot C_b)$. Consequently, no other n-Optimal Schedule consumes less energy than the Original Schedule. ▣

In the n-Optimal Schedule, the total energy consumption of each activated core for the time D is $(C' \cdot \frac{p_m}{f_m} + (\lceil \frac{C}{S[n]} \rceil - C') \cdot \frac{p_{m-1}}{f_{m-1}})$, and its average power consumption (the average energy consumption rate) for the time D is

$$\frac{C' \cdot \frac{p_m}{f_m} + (\lceil \frac{C}{S[n]} \rceil - C') \cdot \frac{p_{m-1}}{f_{m-1}}}{D} \simeq$$

$$P_{m-1} + \frac{p_m - p_{m-1}}{f_m - f_{m-1}} \cdot (\lceil \frac{C}{S[n]} \rceil \cdot \frac{1}{D} - f_{m-1}). \quad (2)$$

In summary, the value of $\lceil \frac{C}{S[n]} \rceil \cdot \frac{1}{D}$ determines the values of f_m and C', and the power consumption of the n-Optimal Schedule. We denote the value of $\lceil \frac{C}{S[n]} \rceil \cdot \frac{1}{D}$ as L_n, and refer to it as Core Load. We also denote the average power consumption of each activated core as a function $P(L_n)$. Then $P(L_n)$ can be formulated as follows:

$$\begin{cases} P_k & \text{if } L_n = f_k \\ P_k + \frac{p_{k+1} - p_k}{f_{k+1} - f_k} \cdot (L_n - f_k) & \text{if } f_k < L_n < f_{k+1} \end{cases} \quad (3)$$

which is a convexly increasing and piece-wisely linear function, as shown in Fig.2.

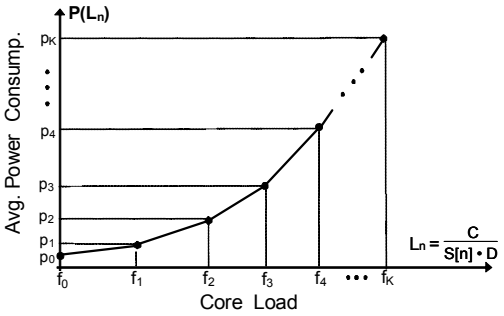

Fig. 2. Average power consumption of an activated core

If $L_n > f_k$, no schedule can complete this task before the deadline. Exploiting the function of $P(L_n)$, we can readily calculate the average power consumption of each n-Optimal Schedule, i.e., $P(L_n) \cdot n$. Comparing the values of $P(L_n) \cdot n$ for $1 \le n \le N$ derives the best schedule with the least energy consumption. The number of activated cores in the best schedule is denoted as η and determined as

$$P(L_\eta) \cdot \eta = \min_{1 \le n \le N} P(L_n) \cdot n \quad . \quad (4)$$

The proposed scheduling, called Minimum-Energy Multi-core Scheduling (MEMS), requires at most one frequency transition per image frame from f_m to f_{m-1} or from f_{m-1} to f_m. If the previous frame completes its execution with f_{m-1}, the next frame starts its execution with f_{m-1} and completes it with f_m in a reverse manner to minimize the frequency transitions.

Ⅳ Evaluation

We compare the proposed off-line scheduling with two greedy methods: single-core scheduling and all-cores scheduling. The single-core scheduling [3-5,7-10] executes the real-time task on a single core and turns off the power of the other cores. The all-cores scheduling [13] executes the task on all available cores. For a fair comparison, these two greedy methods use the respective minimum-energy feasible schedules on a single core and on all cores.

1. Impacts of Task Workload, Available Cores and Speedup

In the first set of comparisons, we examine the performance of the relative computation amount to the deadline and the number of cores available in the system. To measure the relative computation amount to the deadline, we define the ratio of the completion time of given computation cycles under the maximum frequency to the deadline as Task Load, i.e., $\frac{C}{f_k \cdot D} \times 100$. Here we do not consider the time delay and the extra energy required to change the frequency at runtime.

Fig. 3 show the average power consumptions of the three methods, where N denotes the number of all available cores. The speedup model of the MPEG-heavy task is used to evaluate parallel execution. Given Task Load, the single-core scheduling consumes the same power regardless of the number of available cores. The proposed scheduling consumes less power as available cores increase. The values on the top of bars in Fig. 3(b) indicate the number η of cores activated by the proposed scheme, determined in Eq. (4). The all-cores scheduling consumes less power as the number of available cores becomes closer to η but consumes more power as the number of cores becomes larger than η. Compared with the single-core scheduling, the proposed scheduling saves a significant amount of power when the Task Load is heavy. For instance, the power saving ratio of the proposed scheduling is 67% when 'Task Load' = 90% on 4 or more cores. Compared with the all-cores scheduling, the proposed scheduling saves a significant amount of power when the Task Load is light or the number of available cores is large. For instance, the power saving ratio of the proposed scheduling is 89% when 'Task Load' = 10% on 14 cores.

In the second set of comparison, we examine the performance of three different speedup models described in Section Ⅱ-1 : MPEG-light task, Sublinear task, and Concave task. To directly compare the proposed scheduling with the two greedy methods, we measure the ratio of the average power consumption of the proposed scheduling to that of a greedy method, referred to as Normalized Power Consumption (NPC).

Fig. 4(a) and (b) show NPC values against the single-core scheduling and the all-cores scheduling respectively, where β in 'TL=β' denotes the value of Task Load. In Fig. 4(a), NPC values of the three tasks decrease as there are more available cores, but eventually reaches the bound, because $\eta \leq 6$ for $N \geq 6$. The task with a higher speedup shows a smaller NPC than the task with a lower speedup on a given number of cores. If the task has a lower speedup of parallel execution, its completion time becomes closer to the deadline and thus the proposed scheduling has little chance to assign a lower frequency to the activated cores. In Fig. 4(b), NPC values of the three tasks also decrease, as there are more available cores, because η is the same for a large N. Fig. 4 show that the proposed scheduling saves a manifest amount of energy for the tasks with a low speedup of parallel execution.

2. Overhead and Implementation Issues

The operation to exclude all defective frequencies requires O(K) steps. The operation to find the number η of activated cores and the frequency f_m requires O(N·logK) steps. It is a one-time cost, incurred at compilation time. The proposed method requires at most one frequency transition during the execution of each frame. The frequency transition incurs extra energy and time delay. The extra energy is relatively small, compared with

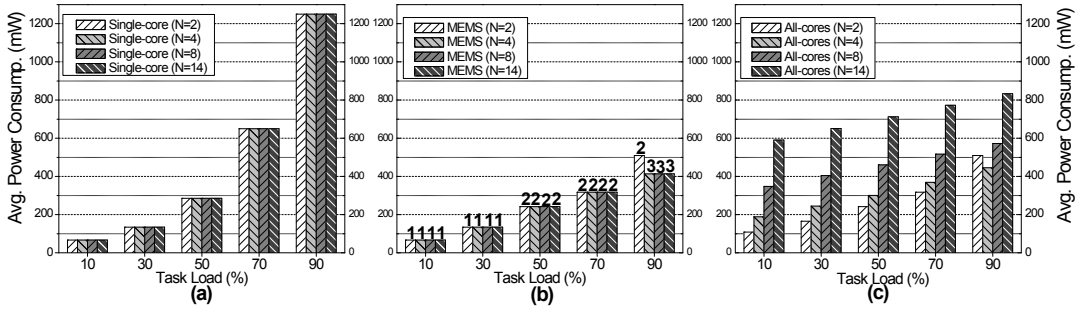

Fig.3. Average power consumptions of (a) the single-core scheduling, (b) the proposed scheduling, and (c) the all-cores scheduling

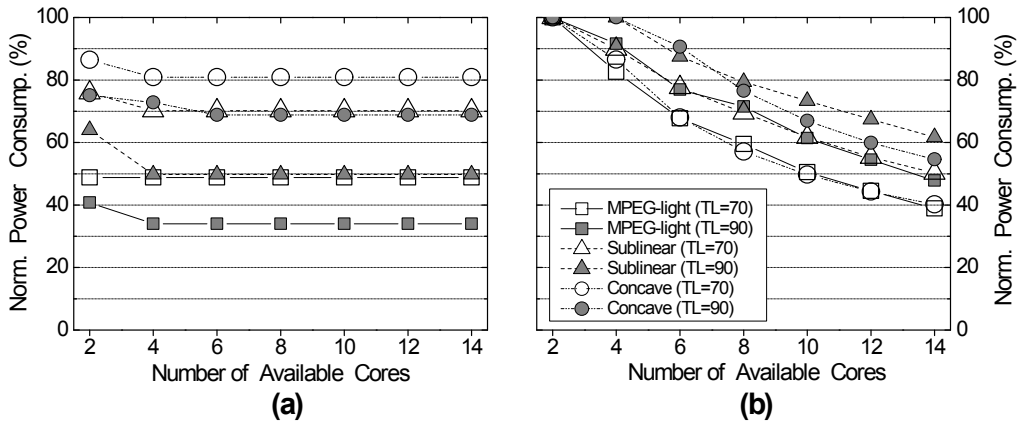

Fig.4. Normalized power consumption of the proposed scheduling against (a) the single-core scheduling and (b) the all-cores scheduling

huge computation amount of each image frame [8]. To accommodate the time delay, denoted as τ. incurred when changing the supplied frequency at runtime, the deadline D is replaced with $D' = D - \tau$. The transition overhead is also required by the single-core scheduling and the all-cores scheduling.

The proposed method needs information about the parallel execution speedup on various numbers of cores, the maximum cycles to be computed within each frame, and the time limit between two adjacent frames in advance. Approximate values of the speedup, computation cycles, and the deadline are obtained from the type of media codec, analysis tool [15] and accumulated statistics [5,8]. Although their worst-case values are adopted for simplicity in this paper, the statistical distribution of varying computation cycles can be exploited for further energy saving [5,8]. This stochastic approach assigns a lower frequency to earlier cycles being executed with a higher probability, and a higher frequency to later cycles being executed with a lower probability. Similar to the MEMS algorithm, it finds feasible schedules consuming the minimum mean energy for parallel executions on each number of allocated cores, and selects the best schedule from the found schedules.

The proposed off-line scheduling minimizes the energy consumption of a periodic real-time task on the DVFS-enabled multi-core platform. Compared with its counterparts assigning either a single core or all cores to the task execution, the proposed scheduling achieves as high as 67% and 89% energy

saving, respectively. As long-lived multi-media tasks such as video playing are becoming increasingly popular, such high potential gain warrants further serious investigation for the emerging multi-core mobile processor.

V Conclusions

The proposed off-line scheduling scheme minimizes the energy consumption of a periodic real-time task on the DVFS-enabled multi-core platform. The proposed scheme activates the best number of cores and inactivates the other unused cores, in order to minimize the energy consumption. Compared with its counterparts assigning either a single core or all cores to the task execution, the proposed scheme achieves as high as 67% and 89% energy saving, respectively. As long-lived multi-media tasks such as video playing are becoming increasingly popular, such high potential gain warrants further serious investigation for the emerging multi-core mobile processor.

In future study, we will investigate an energy-efficient on-line scheme that dynamically makes an adaptive schedule for the periodic real-time task with varying computation amount and minimizes the energy consumption of the multi-core processor while meeting the deadline.

References

[1] L. D. Paulson, ''TV comes to the mobile phone,'' Computer, vol. 39, no. 4, pp. 13-16, 2006.

[2] Semiconductor Industry Association (SIA), International Technology Roadmap for Semiconductors: 2005 Edition, http://www.itrs.net.

[3] J. H. Anderson and S. K. Baruah, ''Energy-efficient synthesis of periodic task systems upon identical multiprocessor platforms,'' Int'l Conf. Distributed Computing Systems, 2004, pp. 428-435.

[4] C.-Y. Yang, J.-J. Chen, and T.-W. Kuo, ''An approximation algorithm for energy-efficient scheduling on a chip multiprocessor,'' Design, Automation and Test in Europe Conf., 2005, pp. 468-473.

[5] R. Xu, C. Xi, R. Melhem, and D. Mosse, ''Practical PACE for embedded systems,'' Int'l Conf. Embedded Software, 2004, pp. 54-63.

[6] A. Bilas, J. Fritts, and J. P. Singh, ''Real-time parallel MPEG-2 decoding in software,'' Int'l Symp. Parallel Processing, 1997, pp. 197-203.

[7] A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. M. A. Hashimi, ''Energy optimization of multiprocessor systems on chip by voltage selection,'' IEEE Trans. Very Large Scale Integr. Syst., vol. 15, no. 3, pp. 262-275, 2007.

[8] W. Yuan and K. Nahrstedt, ''Energy-efficient soft real-time CPU scheduling for mobile multimedia systems,'' ACM Symp. Operating Systems Principles, 2003, pp. 149-163.

[9] H. Kim, H. Hong, H.-S. Kim, J.-H. Ahn, and S. Kang, ''Total energy minimization of real-time tasks in an on-chip multiprocessor using dynamic voltage scaling efficiency metric,'' IEEE Trans. CAD Integr. Circuits Syst., vol. 27, no. 11, pp. 2088-2092, 2008.

[10] E. Seo, J. Jeong, S. Park, and J. Lee, ''Energy efficient scheduling of real-time tasks on multicore processors,'' IEEE Trans. Parallel Distrib. Syst., vol. 19, no. 11, pp. 1540-1552, 2008.

[11] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwereins, ''Energy-aware runtime scheduling for embedded-multiprocessor SOCs,'' IEEE Design & Test of Computers, vol. 18, no. 5, pp. 46-58, 2001.

[12] J. Li and J. F. Martinez, ''Dynamic power-performance adaptation of parallel computation on chip multiprocessors,'' Int'l Symp. High-Performance

Computer Architecture, 2006, pp. 77-87.

[13] W. Y. Lee and H. Lee, ''Energy-efficient scheduling for multiprocessors,'' Electronics Letters, vol. 42, no. 21, pp. 1200-1201, 2006.

[14] L. Benini, A. Bogliolo, and G. D. Micheli, ''A survey of design techniques for system-level dynamic power management,'' IEEE Trans. VLSI Syst., vol. 8, no. 3, pp. 299-316, 2000.

[15] A. Maxiaguine, S. Kunzli, and L. Thiele, ''Workload characterization model for tasks with variable execution demand,'' Design, Automation and Test in Europe Conf., 2004, pp. 1040-1045.

[16] G. No, ''An error control algorithm for wireless video transmission based on feedback channel,'' Journal of the Korea Society of Computer and Information, vol. 7, no. 2, pp. 95-100, 2002.

[17] J. Choi, N. Park, and D. Ahn, ''A lower power scheduling and allocation for multiple supply voltage,'' Journal of the Korea Society of Computer and Information, vol. 7, no. 2, pp. 79-86, 2002.

저 자 소 개

백 형 구

2000 : 부경대학교 전자계산학과 학사
2002 : 부경대학교 전산정보학과 석사
현재 : 부경대학교 컴퓨터공학과 박사과정 및 (주)인텔리코드 개발이사
관심분야 : 클라우드 컴퓨팅, 데이터베이스, EMFG
Email : hgpaek@intellicode.co.kr

여 정 모

1993 : 울산대학교 공학박사
현재 : 부경대학교 컴퓨터공학과 교수 및 (주)엔코아 사외이사
관심분야 : 데이터베이스, ITA/EA, IMS, EMFG
Email : yeo@pknu.ac.kr

이 완 연

2000 : POSTECH 공학박사
2000-2003 : LG전자 선임연구원
2003-2011 : 한림대학교 조교수/부교수
현재 : 동덕여자대학교 컴퓨터학과 부교수
관심분야 : 내장형 컴퓨터, 시스템소프트웨어, 모바일 컴퓨팅
Email : wanlee@dongduk.ac.kr