LETTER *Special Section on Parallel and Distributed Computing and Networking*

Minimum-Energy Semi-Static Scheduling of a Periodic Real-Time Task on DVFS-Enabled Multi-Core Processors*

Wan Yeon LEE[†a)], **Hyogon KIM**[††], ***Nonmembers***, ***and*** **Heejo LEE**[††], ***Member***

SUMMARY The proposed scheduling scheme minimizes the energy consumption of a real-time task on the multi-core processor with the dynamic voltage and frequency scaling capability. The scheme allocates a pertinent number of cores to the task execution, inactivates unused cores, and assigns the lowest frequency meeting the deadline. For a periodic real-time task with consecutive real-time instances, the scheme prepares the minimum-energy solutions for all input cases at off-line time, and applies one of the prepared solutions to each real-time instance at runtime.

key words: *scheduling, real-time task, multi-core, energy, DVFS*

1. Introduction

The limited battery life of mobile devices becomes a burning issue. In the dynamic voltage and frequency scaling (DVFS) mechanism, the processor speed is proportional to the supplied clock frequency and its energy consumption is proportional to a polynomial function of the clock frequency. In order to reduce the energy consumption, state-of-the-art scheduling schemes decrease the supplied clock frequency to the lower bound meeting all deadlines of given real-time tasks. While the energy-efficient real-time scheduling scheme on the multi-core platform has been intensively investigated for *many* real-time tasks [1]–[6], it has been rarely considered for a *single* real-time task due to overabundant hardware platform of the multi-core processor.

Allocating multiple cores to parallel processing of a real-time task whilst lowering the supplied clock frequency can open a rich possibility of reducing the energy consumption. Unfortunately, allocating all available cores to the execution does not directly result in the minimum energy consumption due to two main reasons; One is the non-linear speedup of parallel processing with regard to the number of allocated cores [7]. The other is the irregular energy consumptions of discretely available frequencies in real-life DVFS-enabled processors [3]. In this paper, we propose a scheduling scheme that minimizes the energy consumption of a real-time task by fully exploiting both the parallel processing on multiple cores and the DVFS capability. Considering the non-linear scaling property of parallel processing and the irregular energy consumptions of discrete frequencies, the proposed scheme allocates a pertinent number of cores to the task execution, inactivates 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 virtually generated with a combined use of two adjacent discrete frequencies.

The proposed scheme works in a semi-static manner for a periodic real-time task with consecutive real-time instances; At off-line time, the scheme prepares the minimum-energy schedules for all input cases. When calculating the minimum-energy schedule, the scheme takes into account the extra energy required to activate/inactivate cores. At runtime, the scheme applies one of the prepared schedules to each real-time instance in accordance with the task's input. Evaluation results show that the scheme saves significant energy of the previous method that executes the task on a single core while inactivating the other cores, up to 60% energy consumption of the processing cores. In this paper, the proposed scheme is applied to a real-time video stream running long on the DVFS-enabled multi-core platform, as the video stream represents the most popular and energy-demanding application for current and future mobile devices. Note that the proposed scheme is applicable to other long-lived periodic real-time tasks as well as the video stream.

Numerous approaches have been suggested to save energy of multi-core processors with the DVFS capability. Except a few recent schemes [8], [9], all of the existing approaches [1]–[6]** considered only the case that the number of running real-time tasks is larger than that of available processing cores, but not the opposite case (the number of running real-time tasks is smaller than that of processing cores). Only two off-line scheduling methods [8], [9] exploited overabundant processing units of multi-core processors for the energy-saving parallel processing. These two off-line methods require the runtime information of each real-time task in advance and search for a near minimum-energy schedule, instead of the minimum-energy schedule, due to enormous search space of consecutive real-time tasks with different inputs. In contrast, the main benefit of the proposed scheme is to rapidly find the minimum-energy solutions of all input cases without any runtime information

Manuscript received December 17, 2010.

[†]The author is with Dongduk Women's University, South Korea.

[††]The authors are with Korea University, South Korea.

*This research was supported by Basic Science Research Program through the NRF of Korea funded by the Ministry of Education, Science and Technology (2009-0064347), and by the MKE, Korea, under the ITRC support program supervised by the NIPA (NIPA-2011-C1090-1131-0007).

a) E-mail: wanlee@dongduk.ac.kr

DOI: 10.1587/transinf.E94.D.2389

**Six studies frequently cited are selected among numerous energy-efficient scheduling schemes with the DVFS capability on the multi-core platform.

and store them with a low memory overhead. In accordance with the task's input information obtained at runtime, one of the minimum-energy solutions found at off-line time is applied to each real-time task. Moreover, the proposed scheme is designed to operate over discretely available frequencies with their arbitrary energy consumptions and considers the energy overhead for activating/inactivating cores, whereas the previous methods are designed over infinitely continuous frequencies with an enforced energy consumption formula [1], [2], [4]–[6], [8] and do not consider the energy overhead for activating/inactivating cores [5], [6], [8], [9].

2. System Model and Problem Definition

There are N homogeneous processing cores available in the processor. An identical clock frequency is supplied to all activated cores. Core running speed is proportional to the supplied clock frequency. Available clock frequencies are finite and discrete. Available K discrete frequencies are denoted as $f_1, \cdots, f_K$ in increasing order. For each f_i where $1 \leq i \leq K$, the power consumption is denoted as p_i. Then the energy consumption and execution time of each cycle are $\frac{p_i}{f_i}$ and $\frac{1}{f_i}$, respectively. If $f_i < f_j$, then $p_i < p_j$. The power consumption in the idle status, i.e., leakage power consumption, 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 running speed is semantically equivalent to zero. Unused cores can change their state into the dormant mode (inactivated mode) [6].

A video stream consists of consecutive real-time tasks, i.e., image frames that periodically arrive from the network. The considered system delays a little the execution of an arriving task and buffers it temporarily in order to reduce jitter. The scheduler can then measure the input amounts of the buffered tasks, but not those of un-arrived tasks. The buffered task must be executed within the task arrival period, i.e., the deadline. Each image frame task requires the video decoding operation. The considered system reads ahead the next instruction/data block and thereby hides the memory access latency for a fast video decoding operation. Then the video decoding operation is rarely interrupted by the memory accesses and becomes seriously computation-intensive. The massive decoding computations of image frames can be partitioned into multiple *independent* subtasks, e.g., disjoint partitions of an image frame and separate groups of the image frames [7], and can be executed in parallel on multiple cores. The parallel processing speedup is approximately proportional to the number of allocated cores, but usually smaller than the number of the allocated cores due to *inefficiency factors*, such as extra communications between subtasks and unbalanced completion of subtasks. The parallel processing speedup on n cores is denoted as $S[n]$. In the considered system, the speedup values are obtained empirically and their worst-case speedup is used for safety.

The problem tackled in this article is to minimize the total energy consumption of the processing cores executing a real-time task while meeting the deadline D on N homogeneous cores. If n cores are allocated to the task execution, the other $(N - n)$ cores are inactivated to save energy. The task has C^{tot} computation cycles that must be completed within the deadline D. When the C^{tot} computation cycles are executed in parallel on n cores with a speedup of $S[n]$, they can be completed within at most $\lceil \frac{C^{tot}}{S[n]} \rceil$ cycles. The computation amount C^{tot}, the deadline D, and the speedup values $S[n]$ for $1 \leq n \leq N$ are given to the scheduler. A schedule is referred to as *n-feasible* if it allocates n cores and completes the task within the deadline D. An n-feasible schedule is called *n-Optimal Schedule* if it consumes the minimal energy among all n-feasible schedules.

3. Proposed Scheduling Scheme

The scheme first excludes the *energy-inefficient* frequency f_y such that $\frac{p_y - p_x}{f_y - f_x} > \frac{p_z - p_y}{f_z - f_y}$ for some $f_x < f_y < f_z$. Based on the following Lemma 1, the energy-inefficient frequency is discarded henceforth. Calculating $\frac{p_k - p_{k-1}}{f_k - f_{k-1}} > \frac{p_{k+1} - p_k}{f_{k+1} - f_k}$ for $1 < k < K$ can select all energy-inefficient frequencies.

Lemma 1: If $\frac{p_y - p_x}{f_y - f_x} > \frac{p_z - p_y}{f_z - f_y}$ for some $f_x < f_y < f_z$, any n-Optimal Schedule does not use f_y.

proof: See our preliminary study [10]. □

Next, the scheme prepares the minimum-energy schedules for all values of C^{tot}. It finds each n-Optimal Schedule for all C^{tot}s. Among the found n-Optimal Schedules, it selects the best n-Optimal Schedule with the least energy consumption for the individual value of C^{tot}. The n-Optimal Schedule is found as follows; It chooses a frequency f_m that is nearest to and no smaller than $\left\lceil \frac{C^{tot}}{S[n]} \right\rceil \cdot \frac{1}{D}$ from $f_1, \cdots, f_K$. If $\left\lceil \frac{C^{tot}}{S[n]} \right\rceil \cdot \frac{1}{f_m} = D$, the scheme assigns f_m to the C^{tot} cycles. In case of $\left\lceil \frac{C^{tot}}{S[n]} \right\rceil \cdot \frac{1}{f_m} < D$, the scheme assigns f_m and f_{m-1}. To find the transition point C' between f_m and f_{m-1}, it solves an equation $\frac{C'}{f_m} + \frac{\left\lceil \frac{C^{tot}}{S[n]} \right\rceil - C'}{f_{m-1}} = D$. Then

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

The following Theorem 1 verifies that the n-Optimal Schedule assigns f_m to $\lceil C' \rceil$ cycles and f_{m-1} to the remaining $\left(\left\lceil \frac{C^{tot}}{S[n]} \right\rceil - \lceil C' \rceil \right)$ cycles.

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

proof: See our preliminary study [10]. □

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

$$\frac{C' \cdot \frac{p_m}{f_m} + \left(\left\lceil \frac{C^{tot}}{S[n]} \right\rceil - C'\right) \cdot \frac{p_{m-1}}{f_{m-1}}}{D} \simeq p_{m-1} + \frac{p_m - p_{m-1}}{f_m - f_{m-1}} \cdot \left(\left\lceil \frac{C^{tot}}{S[n]} \right\rceil \cdot \frac{1}{D} - f_{m-1}\right). \quad (2)$$

The value of $\lceil \frac{C^{tot}}{S[n]} \rceil \cdot \frac{1}{D}$ determines f_m, C' and the power consumption of the n-Optimal Schedule.

The normalized value of $\lceil \frac{C^{tot}}{S[n]} \rceil \cdot \frac{1}{D}$ to the maximum frequency, $\lceil \frac{C^{tot}}{S[n]} \rceil \cdot \frac{1}{D \cdot f_K}$, is referred to as *Core Load* and denoted as L_n. Also, the ratio of the completion time of a task under the maximum frequency to the deadline, $\frac{C^{tot}}{D \cdot f_K}$, is referred to as *Task Utilization* and denoted as U. Because $L_n = \frac{U}{S[n]}$, $L_n = U$ if $n = 1$. As n increases, L_n becomes much smaller than U. Then the average power consumption of each activated core in the n-Optimal Schedule can be formulated as a function $P(L_n)$ because it is determined by the value of L_n. From Eq. (2), $P(L_n)$ is

$$\begin{cases} p_i & \text{if } L_n = \frac{f_i}{f_K} \\ p_i + \frac{p_{i+1} - p_i}{\frac{f_{i+1}}{f_K} - \frac{f_i}{f_K}} \cdot (L_n - f_i) & \text{if } \frac{f_i}{f_K} < L_n < \frac{f_{i+1}}{f_K} \end{cases}$$

which is a *convex* and *piece-wise linear* function. If $L_n > 1$, no schedule can complete this task before the deadline. Exploiting the $P(L_n)$ function, we can readily calculate the mean power consumption of each n-Optimal Schedule. When energy overhead for activating and inactivating cores is not considered, the number of activated cores in the minimum-energy schedule is determined as

$$\min_{1 \le n \le N} \{ P(L_n) \cdot n \cdot D + p_{dor} \cdot (N - n) \cdot D \} \quad (3)$$

where p_{dor} is the power consumption in the dormant mode.

Now, let us consider the extra energy for activating and inactivating cores. The extra energy, denoted as Δ, depends on the number of currently activated cores. When the number of currently activated cores is N_{act}, the extra energy required for the n-Optimal Schedule is calculated as

$$\Delta = \begin{cases} E_{act} \cdot (n - N_{act}) & \text{if } n > N_{act} \\ E_{ina} \cdot (N_{act} - n) & \text{otherwise,} \end{cases}$$

where E_{act} and E_{ina} are the activating and the inactivating energies of a core, respectively. Then Eq. (3) is replaced with the following formula:

$$\min_{1 \le n \le N} \{ P(L_n) \cdot n \cdot D + p_{dor} \cdot (N - n) \cdot D + \Delta \}. \quad (4)$$

The solution to Eq. (4) depends on $L_n = \frac{U}{S[n]}$ and N_{act}.

For each N_{act}, the scheme separately derives and manages the solution to Eq. (4). Given a fixed N_{act}, the scheme finds all n-Optimal Schedules for $0 \le L_n \le 1$ (or $0 \le U \le 1$). In order to obtain each n-Optimal Schedule, at most $(K + 1)$ input values of L_n, *i.e.*, $\frac{f_0}{f_K}, \cdots, \frac{f_K}{f_K}$, are calculated. It is unnecessary to calculate the rest of the L_n values because $P(L_n)$ is a linear function where $\frac{f_i}{f_K} < L_n < \frac{f_{i+1}}{f_K}$. The scheme compares the energy consumptions of all the n-Optimal Schedules with regard to $U = L_n \cdot S[n]$ and selects the best n-Optimal Schedule with the least energy consumption. The selected minimum-energy schedule is a piece-wise linear function of U because all the n-Optimal Schedules are piece-wise linear functions of U. In a linear range, the n and f_m values of the minimum-energy schedule are fixed, yet the C' value varies in accordance with U. The scheme calculates the start and end U values of the linear range and stores its corresponding n and f_m values into a single range bin of U. This procedure is performed separately for each N_{act}. A working example will be given in the next section.

If we know only the n and f_m values of the minimum-energy schedule, we can calculate the transition point C' with a given $U = \frac{C^{tot}}{D \cdot f_K}$ by using Eq. (1), and automatically generate the schedule assigning f_m to $\lceil C' \rceil$ cycles and f_{m-1} to $\left(\left\lceil \frac{C^{tot}}{S[n]} \right\rceil - \lceil C' \rceil\right)$ cycles. At runtime, one of the pre-computed schedules is applied to each real-time task dynamically arriving, in accordance with the given U and N_{act}.

4. Evaluation

The proposed scheme is compared with the previous method that executes a task on a single core [1]–[6]. For the sake of fairness, we assume that the previous method also inactivates unused cores and generates the 1-Optimal Schedule with a combined use of two discrete frequencies, although it did not actually consider inactivating unused cores and was designed over infinitely continuous frequencies. Figure 1 shows the evaluation model. Figure 1 (a) shows five available frequencies and their power consumptions (energy consumption rates) obtained from a well-known DVFS processor, the Intel XScale [3]. The values of p_0, p_{dor}, E_{act} and

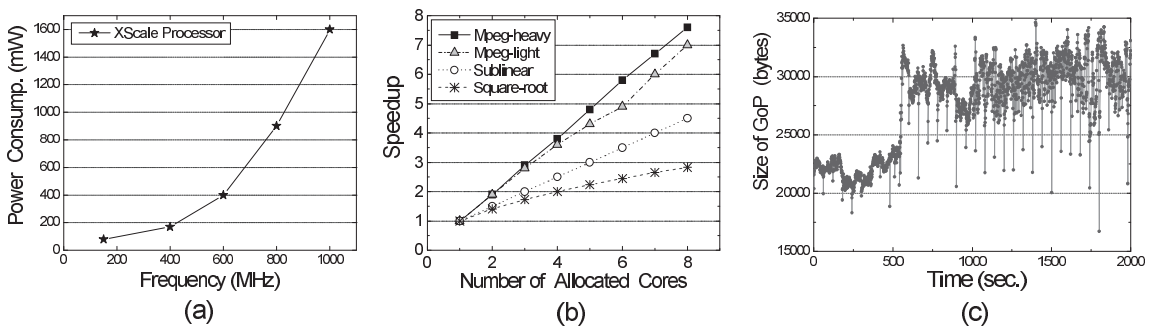

Fig. 1 Evaluation model: (a) DVFS processor, (b) speedup of parallel processing and (c) video stream.

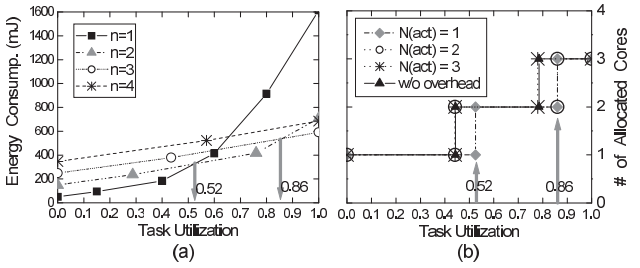

Fig. 2 (a) n-Optimal Schedules and (b) the number of cores allocated to the best schedule.

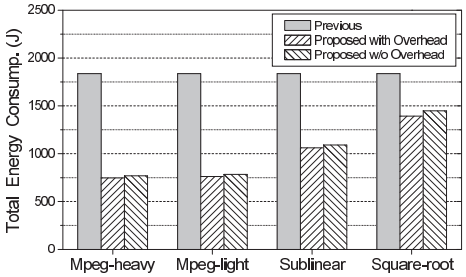

Fig. 3 Energy consumptions against four speedup models.

E_{ina} are set on the basis of 50 nm or 70 nm CMOS technology [6], [11]; p_0 is 44% of p_1, p_{dor} is 3% of p_0, E_{act} is 64 mJ and E_{ina} is 36 μJ. Figure 1 (b) shows four speedup models of parallel processing speedups. Mpeg-heavy and Mpeg-light speedup models are drawn from the parallel MPEG-2 video task on the Silicon Graphics Challenge multiprocessor [7]. Mpeg-heavy and Mpeg-light are the results of 1408×960 and 352×240 resolution video tasks, respectively. Sublinear and Square-root speedup models are synthetically generated to represent lower parallel processing speedup. Sublinear speedup is set with $S[n] = (n-1) \times 0.5 + 1$. Square-root speedup is set with $S[n] = \sqrt{n}$ for $n \geq 1$. N is set to 8. Figure 1 (c) shows the Group of Pictures (GoP) sizes of a video stream applied to our evaluation. The video stream is the encoded image frames of captured real-life scenes. Each GoP is generated for a second. Scheduler is invoked per a second and provides a schedule for each GoP in the buffer, where the deadline of each GoP is one second. The U value of a GoP with 35,000 bytes is set to 1.

Figure 2 (a) shows the energy consumptions of n-Optimal Schedules for the XScale DVFS processor, where $N_{act} = 1$ and Mpeg-heavy speedup is used. The n-Optimal Schedules only when $n \leq 4$ are displayed, because those when $n > 4$ have larger energy consumption. As the best schedule, the scheduler selects the 1-Optimal Schedule where $0 \leq U \leq 0.52$, the 2-Optimal Schedule where $0.52 < U \leq 0.86$, and the 3-Optimal Schedule where $0.86 < U \leq 1$. Their n and f_m values are stored into seven range bins of U: $\{1, f_1\}$ in $\left(\frac{f_0}{f_5}, \frac{f_1}{f_5} \right]$, $\{1, f_2\}$ in $\left(\frac{f_1}{f_5}, \frac{f_2}{f_5} \right]$, $\{1, f_3\}$ in $\left(\frac{f_2}{f_5}, 0.52 \right]$, $\{2, f_2\}$ in $\left(0.52, \frac{f_1}{f_5} \cdot S[2] \right]$, $\{2, f_3\}$ in $\left(\frac{f_1}{f_5} \cdot S[2], 0.86 \right]$, and $\{3, f_2\}$ in $(0.86, 1.0]$. Figure 2 (b) shows the number of cores allocated to the minimum-energy schedule versus Task Utilization U. 'N(act)' denotes N_{act}. 'w/o overhead' denotes the result when neglecting the extra energy for activating and inactivating cores. Figure 2 (b) shows that the extra energy affects the decision on the best schedule.

Figure 3 shows the total energy consumptions of the video stream when the four speedup models shown in Fig. 1 (b) are used separately. 'Previous' denotes the result of the previous method, and 'Proposed with Overhead' denotes that of the proposed scheme. The proposed scheme saves more energy of the previous method for higher parallel processing speedup. The energy saving ratios over the previous method are about are 60% for Mpeg-heavy, 58% for Mpeg-light, 42% for Sublinear, and 24% for Square-root. 'Proposed w/o Overhead' denotes the energy consumption of the proposed scheme when neglecting the extra energy for activating and inactivating cores, i.e., using Eq. (3) instead of Eq. (4). The energy consumption of 'Proposed w/o Overhead' is slightly larger than that of 'Proposed with Overhead' by about 4% increase on average.

References

[1] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," ACM SOSP, pp.149–163, 2003.

[2] J.H. Anderson and S.K. Baruah, "Energy-efficient synthesis of periodic task systems upon identical multiprocessor platforms," ICDCS, pp.428–435, 2004.

[3] R. Xu, C. Xi, R. Melhem, and D. Mossé, "Practical PACE for embedded systems," EMSOFT, pp.54–63, 2004.

[4] C.Y. Yang, J.J. Chen, and T.W. Kuo, "An approximation algorithm for energy-efficient scheduling on a chip multiprocessor," DATE, pp.468–473, 2005.

[5] 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. (VLSI) Syst., vol.15, no.3, pp.262–275, 2007.

[6] 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.

[7] A. Bilas, J. Fritts, and J.P. Singh, "Real-time parallel MPEG-2 decoding in software," IPPS, pp.197–203, 1997.

[8] W.Y. Lee and H. Lee, "Energy-efficient scheduling for multiprocessors," Electron. Lett., vol.42, no.21, pp.1200–1201, 2006.

[9] J. Li and J.F. Martínez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," HPCA, pp.77–87, 2006.

[10] W.Y. Lee, H. Kim, and H. Lee, "Energy-efficient scheduling of a real-time task on DVFS-enabled multi-cores," tech. rep., Korea University, CIC-CCS-TR 08-001, 2008.

[11] L. Benini, A. Bogliolo, and G.D. Micheli, "A survey of design techniques for system-level dynamic power management," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.8, no.3, pp.299–316, 2000.