



HAL
open science

An Approximation Algorithm for Energy-Efficient Scheduling on A Chip Multiprocessor

Chuan-Yue Yang, Jian-Jia Chen, Tei-Wei Kuo

► **To cite this version:**

Chuan-Yue Yang, Jian-Jia Chen, Tei-Wei Kuo. An Approximation Algorithm for Energy-Efficient Scheduling on A Chip Multiprocessor. DATE'05, Mar 2005, Munich, Germany. pp.468-473. hal-00181556

HAL Id: hal-00181556

<https://hal.science/hal-00181556>

Submitted on 24 Oct 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An Approximation Algorithm for Energy-Efficient Scheduling on A Chip Multiprocessor*

Chuan-Yue Yang, Jian-Jia Chen, and Tei-Wei Kuo
Department of Computer Science and Information Engineering,
Graduate Institute of Networking and Multimedia,
National Taiwan University, Taipei, Taiwan, ROC.
Email: {r92032, r90079, ktw}@csie.ntu.edu.tw

Abstract

In the recent decade, voltage scaling has become an attractive feature for many system component designs. In this paper, we consider energy-efficient real-time task scheduling over a chip multiprocessor architecture. The objective is to schedule a set of frame-based tasks with the minimum energy consumption, where all tasks are ready at time 0 and share a common deadline. We show that such a minimization problem is NP-hard and then propose a 2.371-approximation algorithm. The strength of the proposed algorithm was demonstrated by a series of simulations, for which near optimal results were obtained.

1 Introduction

With the increasing popularity and prevailing supports on voltage scaling for electronic circuits, energy efficiency has become a highly important design issue in hardware and software implementations [6, 10, 13, 17]. The design of electronic circuitry is usually done such that a higher supply voltage would result in a higher execution speed (or a higher frequency). An example power consumption function [3, 18], as follows, shows the power consumption of a processor as a function of the processor speed:

$$P(s) = C_{ef}V_{dd}^2s \quad (1)$$

where $s = k_h((V_{dd} - V_t)^2)/V_{dd}$, and $P, s, C_{ef}, V_t, V_{dd}$, and k_h denote the power consumption, the processor speed, the effective switch capacitance, the threshold voltage, the supply voltage, and a hardware-design-specific constant, respectively ($V_{dd} \geq V_t \geq 0, k_h > 0$, and $C_{ef} > 0$).

Modern superscalar processors achieve excellent performance through pipelined instruction executions and concurrent services of independent instruction streams. Further

performance improvement is often done by increasing the processor frequency but at the cost of energy consumption. When energy consumption and system performance must be considered at the same time, the multiprocessor architecture seems being a reasonable choice, especially for multiprogramming environments (e.g., [16]). The chip-multiprocessor architecture is proposed as an attempt to overcome the chip-space constraint and the processor communication delay problem for the multiprocessor architecture. With a slight increasing on the die size, multiple processors, i.e., cores, are mounted on a single die to retain many advantages of the multiprocessor architecture but with only comparatively low wire delay. While many excellent research results have been proposed for uniprocessor energy-efficient scheduling, e.g., [2, 5, 12, 14, 19, 20], little work has been done for multiprocessor systems, e.g., [4, 8, 21], even though the multiprocessor architecture has become increasingly popular in various platforms. The strong demand for multiprocessor energy-efficient scheduling is not only from server systems but also from the embedded systems, such as System-on-Chip systems. In particular, power saving on the chip multiprocessor architecture received much attention recently, especially due to its applications to embedded systems. For example, the adaptive chip-multiprocessor (ACMP) architecture proposed in [16] allows each core to switch its operation mode among RUN, STANDBY, and DORMANT in a dynamic manner to reduce the energy consumption.

Real-time task scheduling over a chip multiprocessor with the capability of dynamic voltage scaling (abbreviated as DVS-CMP) is exploited in this paper. The DVS-CMP architecture explored in this paper has M homogeneous cores, where each core could be dormant independently, and all non-dormant cores must operate at the same voltage supply. The objective of this research is to schedule a set of frame-based tasks with the minimum energy consumption on a DVS-CMP processor, where all tasks are ready at time 0 and share a common deadline. We show that such a scheduling problem is NP-hard and propose a 2.371-approximation algorithm. The strength of the proposed algorithm is demonstrated by a series

*Support in parts by research grants from ROC National Science Council (NSC-93-2213-E-002-031).

of simulations, for which we have near optimal results.

The rest of this paper is organized as follows: Section 2 presents related work on energy-efficient scheduling. In Section 3, we formally define the problem under considerations. An approximation algorithm was proposed for energy-efficient scheduling in Section 4. Simulation results are shown in Section 5 to evaluate the capability of the proposed algorithm. Section 6 is the conclusion.

2 Related Work

Energy-efficient scheduling has been an active research topic in the past decade. Although many excellent results have been proposed for uniprocessor real-time task scheduling, e.g., [2, 5, 12, 14, 19, 20], little work is done for multiprocessor systems so far. However, with the strong raising of the market for various multiprocessor architectures and their variations, multiprocessor energy-efficient scheduling has started receiving much attention in recent years, e.g., [1, 4, 8, 9, 21, 15, 22]. In particular, Chen, et al. [4] proposed an approximation algorithm for multiprocessor energy-efficient scheduling over a set of independent frame-based tasks, where all tasks share the same deadline. Gruian proposed a simulated annealing (SA) approach and a list-based heuristic algorithm with a dynamic priority assignment policy for the considerations of precedence constraints [8, 9]. Mishra, et al. [15] explored scheduling issues over the communication delay for tasks. Zhu, et al. [22] explored on-line scheduling for a set of independent/dependent frame-based tasks. Given an off-line schedule with the worst-case task execution times, on-line strategies were proposed to reclaim the slacks resulted from the early completion times of tasks.

Although some research work has been proposed for multiprocessor energy-efficient scheduling, many previous results considered multiprocessor scheduling in which each processor can operate independently at its own processor speed. In this research, we are interested in a DVS-CMP architecture, in which there are M given homogeneous cores, where each core could be dormant independently, and all non-dormant cores must operate at the same voltage supply. Our objective is to schedule a set of frame-based tasks with the minimum energy consumption on the given DVS-CMP processor, where all tasks are ready at time 0 and share a common deadline. The work done by Anderson and Baruah [1] is related to our research in this paper. They proposed algorithms for the synthesizing of a multiprocessor hard-real-time system with independent periodic tasks and exploited the trade-offs between the number of processors in the system and the energy consumption. Different from their work, we aim at scheduling for the energy consumption minimization on a DVS-CMP processor with a fixed number of cores.

3 Problem Definition

In this paper, we exploit energy-efficient scheduling on a chip multiprocessor equipped with M homogeneous cores. The power consumption function in [3, 18], i.e., Formula 1, is adopted in this paper, where $V_t = 0$, or $V_{dd} \gg V_t$. The power consumption function can be rephrased as $P(s) = \alpha s^3$, where α is a constant. The available processor speeds for the DVS-CMP under considerations are assumed being adjustable in a continuous manner, and no upper bound on the processor speed is given (that is as the same as that in [11, 19]). Furthermore, let the overheads on the switching of the supply voltage be negligible. Suppose that any of the cores could be turned into a sleep mode (i.e., $s = 0$) at any time, but all of non-sleeping cores must operate at the same processor speed. Let the energy consumed for a core at the processor speed s for t time units be $P(s)t$, and the execution of an amount c (in cycles on a core) of computation at the processor speed s take c/s time units. The energy-efficient scheduling problem to be explored in this paper could be defined as follows:

Definition 1 *Energy Consumption Minimization for DVS-CMP Scheduling (ECMS):*

Consider a set T of independent tasks on a DVS-CMP, where all tasks in T are ready at time 0 and share a common deadline D . Let each task $\tau_i \in T$ be associated with an amount c_i (in cycles on a core) of computation requirements. The objective of this problem is to minimize the energy consumption in the scheduling of tasks in T without violating the common deadline D , where task migration between any two cores is not permitted.

A *schedule* of an input instance for the ECMS problem is a mapping of the executions of the tasks in the set to cores on the DVS-CMP with speed assignments and their corresponding time intervals. A schedule is *feasible* if no task misses the common deadline D , and the DVS-CMP constraints are not violated. Let $\Phi(\Psi)$ denote the energy consumption of a schedule Ψ . A schedule is *optimal* if it is feasible and its energy consumption is the minimum energy consumption of all feasible schedules.

4 The Proposed Algorithm

4.1 Energy Consumption Minimization with Tasks Being Partitioned

In this subsection, we shall propose a scheduling algorithm in energy consumption minimization when tasks are already partitioned. A *task assignment* is defined as a partition of T into M disjoint subsets $X \stackrel{\text{def}}{=} \{X_1, X_2, \dots, X_M\}$. A schedule that is based on X must have tasks in $X_i \in X$ running on the i -th core. An *optimal* X -based schedule is a schedule that has the minimum energy consumption among

X -based schedules. The load \mathbb{X}_i of each $X_i \in X$ is defined as the total amount of the computation requirements of the tasks in X_i , and the load distribution \mathbb{X} of X is denoted as $\{\mathbb{X}_1, \mathbb{X}_2, \dots, \mathbb{X}_M\}$. Without losing the generality, let $X_i \in X$ be sorted in a non-decreasing order of their loads. For the simplicity of discussions, let $\mathbb{X}_{M+1} = \infty$ and $\mathbb{X}_0 = 0$. A schedule Ψ satisfies the *deep sleeping property* if a core μ is in the sleep mode at any time t' for $t < t' < D$ when μ is found in the sleep mode at some time $0 \leq t < D$.

Lemma 1 *For any task assignment X , there exists an optimal X -based schedule that satisfies the deep sleeping property.*

Proof. Given a feasible X -based schedule ψ that does not satisfy the deep sleeping property, let the time interval $(0, D]$ of ψ be divided into disjoint time fragments such that the processor speeds of non-sleeping cores are different between any two different consecutive time fragments. (Note that the processor speeds of all non-sleeping cores are the same in each fragment by the definitions of DVS-CMP discussed in this paper). Let time fragments that have the same processor speed be merged together such that $(0, D]$ have k fragments, and each t_i of the fragments has a processor speed s_i . Without losing of generality, we let $0 < s_1 < s_2 < \dots < s_k$. Let $y_{i,j}$ denote the number of core cycles needed for the j -th core of the i -th fragment, and $|t_i|$ denote the total duration of the i -th fragment (note that each fragment might consist of non-consecutive time intervals). $\hat{\psi}$ is a schedule derived from ψ as follows: The j -th core executes tasks at the speed s_i in $(\sum_{h=1}^{i-1} |t_h|, \sum_{h=1}^{i-1} |t_h| + y_{i,j}/s_i]$ for $i = 1, 2, \dots, k$ and is turned into the sleep mode in $(\sum_{h=1}^{i-1} |t_h| + y_{i,j}/s_i, \sum_{h=1}^i |t_h|]$ for $i = 1, 2, \dots, k$. Schedules ψ and $\hat{\psi}$ have the same energy consumption.

We shall turn the resulted schedule $\hat{\psi}$ into another schedule ψ' that satisfies the deep sleeping property: Let \hat{t} be the earliest time moment that some core μ_j goes into the sleep mode in $\hat{\psi}$. Let n be the index which satisfies $\sum_{h=1}^{n-1} |t_h| \leq \hat{t} < \sum_{h=1}^n |t_h|$. Suppose that μ_j is non-sleeping at some time moment t' , where $\hat{t} < t' < D$. Let Y be the total number of cycles executed after t' on the core μ_j (for all fragments). These Y cycles on μ_j are then executed in the fragments starting from t_n at their corresponding speeds (t_{n+1} , etc) from the time moment \hat{t} until all of the Y cycles are done. The amount of consumed energy is less in the transformation, due to the convexity of the power consumption function. By repeating the same process for every core, we can always transform a feasible X -based schedule into one that satisfies the deep sleeping property and consumed no more energy than the original one does. \square

Based on the deep sleeping property, we can derive an optimal schedule based on a given task assignment X , as shown in Algorithm 1. Steps 4-7 follow the definition of the power consumption function. The time complexity of Algorithm 1 is $O(|T| + M)$. The optimality is shown as follows.

Algorithm 1 : MES

Input: (X) ;

Output: A feasible X -based schedule Ψ with the minimum energy consumption;

- 1: $\mathbb{X}_i \leftarrow 0$ for $i = 0$ to M ;
 - 2: **for** $i = 1$ to M **do**
 - 3: $\mathbb{X}_i \leftarrow \mathbb{X}_i + c_j$ for $\forall \tau_j \in X_i$;
 - 4: $L \leftarrow \sum_{i=1}^M (\mathbb{X}_i - \mathbb{X}_{i-1}) \sqrt[3]{M-i+1}$ and $t_0 \leftarrow 0$;
 - 5: **for** $i = 1$ to M **do**
 - 6: $s_i \leftarrow \frac{L}{\sqrt[3]{M-i+1}}$ and $t_i \leftarrow t_{i-1} + D \frac{(\mathbb{X}_i - \mathbb{X}_{i-1}) \sqrt[3]{M-i+1}}{L}$;
 - 7: let Ψ turn the i -th core into the sleep mode at t_i and set the speed as s_i in $(t_{i-1}, t_i]$ for the non-sleeping cores;
 - 8: **return** Ψ by executing tasks assigned to each core in an arbitrary order;
-

Lemma 2 *For any given task assignment X , the X -based schedule derived by Algorithm 1 is optimal.*

Proof. Let the energy consumption for the schedule derived by Algorithm 1 be E^* , where $E^* = \frac{\alpha}{D^2} (\sum_{i=1}^M (\mathbb{X}_i - \mathbb{X}_{i-1}) \sqrt[3]{M-i+1})^3$. We shall show that any X -based schedule Ψ (that satisfies the deep sleeping property) consumes no less energy than E^* . Let z_0 be the index such that \mathbb{X}_j is equal to 0 for all $z_0 \geq j \geq 0$. z_i is recursively defined as the index such that \mathbb{X}_j is equal to $\mathbb{X}_{z_{i-1}+1}$ for all $z_i \geq j > z_{i-1}$ and $i \geq 1$. Besides, let k the index such that $z_k = M$. y_i is defined to be $\mathbb{X}_{z_i} - \mathbb{X}_{z_{i-1}}$ for all $i \geq 1$.

Let β_i be the time instant when the $(z_{i-1} + 1)$ -th core is turned into sleep ($\beta_0 = 0$) for Ψ . Due to the deep sleeping property, there are $(M - z_{i-1})$ cores that are non-sleeping in $(\beta_{i-1}, \beta_i]$, and z_{i-1} cores are sleeping in $(\beta_{i-1}, \beta_i]$. Note that X_i 's $\in X$ are sorted in a non-decreasing order of their loads. γ_i is defined as $\beta_i - \beta_{i-1}$. Because of the convexity of the power consumption function, executing y_i cycles at the same speed $\frac{y_i}{\gamma_i}$ for γ_i time units is the best choice for energy consumption. Therefore, $\Phi(\Psi) \geq \alpha \sum_{i=1}^k (M - z_{i-1}) (\frac{y_i}{\gamma_i})^3 \gamma_i$. Furthermore, there must exist at least one core that is not in the sleeping mode before D unless there is no load for any core (because the speed lower bound of each core is 0). Let $\hat{\Phi}(X)$ be the energy consumption of an optimal X -based schedule. We have

$$\hat{\Phi}(X) \geq \min_{\sum_{i=1}^k \gamma_i = D} \alpha \sum_{i=1}^k (M - z_{i-1}) \left(\frac{y_i}{\gamma_i}\right)^3 \gamma_i \quad (2)$$

By adopting the Lagrange multiplier method, the right-hand side of Equations (2) is minimized when

$$\gamma_i = D \cdot \frac{\sqrt[3]{M - z_{i-1}} y_i}{\sum_{j=1}^k \sqrt[3]{M - z_{j-1}} y_j}$$

We have¹

$$\begin{aligned}\Phi(\Psi) &\geq \hat{\Phi}(X) \geq \frac{\alpha}{D^2} \left(\sum_{i=1}^k y_i \sqrt[3]{M - z_{i-1}} \right)^3 \\ &= \frac{\alpha}{D^2} \left(\sum_{i=1}^M (\mathbb{X}_i - \mathbb{X}_{i-1}) \sqrt[3]{M - i + 1} \right)^3 = E^* \geq \hat{\Phi}(X). \quad (3)\end{aligned}$$

Since $E^* = \hat{\Phi}(X)$, we reach the conclusion. \square

Let \mathbb{X} be any given load distribution of T . A schedule is based on \mathbb{X} if the number of cycles executed on the i -th core is equal to \mathbb{X}_i . Let the minimum energy consumption among schedules based on \mathbb{X} be $\hat{\Phi}(\mathbb{X})$. We have the following corollary:

Corollary 1 $\hat{\Phi}(\mathbb{X}) = \frac{\alpha}{D^2} \left(\sum_{i=1}^M (\mathbb{X}_i - \mathbb{X}_{i-1}) \sqrt[3]{M - i + 1} \right)^3$.

4.2 A 2.371-Approximation Algorithm

We shall first show the NP-hardness of the ECMS problem and then propose a 2.371-approximation algorithm.

Corollary 2 *The ECMS problem is NP-hard.*

Proof. Based on Equation (3) in Lemma 2, $\Phi(\Psi)$ is minimum if and only if $\mathbb{X}_j = \frac{\sum_{\tau_i \in T} c_i}{M}$ for $j = 1, 2, \dots, M$. This problem could be reduced from the multiprocessor scheduling problem [SS8] in [7], that is NP-complete. \square

The proposed 2.371-approximation algorithm (Algorithm LTF), as shown in Algorithm 2, adopts the *Largest-Task-First* strategy to partition T into M disjoint sets. Tasks are considered in a non-increasing order of their computation requirements. For the simplicity of discussions, let T be sorted in a non-increasing order of the computation requirements of tasks (where ties could be broken arbitrarily).

Algorithm 2 : LTF

Input: (T, D, M) ;

Output: A feasible schedule Ψ^{LTF} with minimal energy consumption;

- 1: sort all tasks in a non-increasing order of the computation requirements of tasks;
 - 2: $X_i \leftarrow \phi$ and $\mathbb{X}_i \leftarrow 0$ for $i = 1$ to M ;
 - 3: **for** $i = 1$ to $|T|$ **do**
 - 4: find the smallest \mathbb{X}_m ; (break ties arbitrarily)
 - 5: $X_m \leftarrow X_m + \{\tau_i\}$ and $\mathbb{X}_m \leftarrow \mathbb{X}_m + c_i$;
 - 6: reorder X_i by a non-decreasing order of their loads and let $X^{LTF} \leftarrow \{X_1, X_2, \dots, X_M\}$;
 - 7: return the resulted schedule Ψ^{LTF} by applying $\text{MES}(X^{LTF})$;
-

Let T, D , and M denote the task set under discussions, its common deadline, and the number of cores, respectively. Algorithm LTF always assigns a task to the core with the smallest load, where tasks are picked up in a non-increasing order of their computation requirements. The seeking of the core with the smallest load could be done by the manipulation of

¹The detail proof is omitted due to the space limitation.

a heap data structure. The time complexity of Algorithm LTF is $O(|T|(\log |T| + \log M) + M)$, which is dominated by the cost for task sorting and heap manipulation.

Given a task set T (with D as the common deadline) and the number M of cores, Ψ^{LTF} , X^{LTF} , and \mathbb{X}^{LTF} denote the schedule, the task assignment, and the load distribution derived by Algorithm LTF, respectively. For the simplicity of discussions, let us renumber cores such that elements in X^{LTF} be sorted in a non-decreasing order of their loads. That is, $\mathbb{X}_i^{LTF} \leq \mathbb{X}_{i+1}^{LTF}$ for $1 \leq i < M$, where X_i^{LTF} denotes the i -th element in X^{LTF} , and \mathbb{X}_i^{LTF} is the load of X_i^{LTF} . For the abbreviation, \mathbb{X}_i^{LTF} is also referred to as p_i .

Lemma 3 *Given two load distributions \mathbb{X} and \mathbb{X}' for the same task set, $\hat{\Phi}(\mathbb{X}) < \hat{\Phi}(\mathbb{X}')$ if there exist two indices i and j ($j > i$) such that $\mathbb{X}_k = \mathbb{X}'_k$ for $k \neq i, j$, and $0 < \mathbb{X}_i - \mathbb{X}'_i < \min\{\mathbb{X}_i - \mathbb{X}_{i-1}, \mathbb{X}_{j+1} - \mathbb{X}_j\}$.*

Proof. Based on Corollary 1, $\hat{\Phi}(\mathbb{X}) = \frac{\alpha}{D^2} \left(\sum_{k=1}^M (\mathbb{X}_k - \mathbb{X}_{k-1}) \sqrt[3]{M - k + 1} \right)^3$ and $\hat{\Phi}(\mathbb{X}') = \frac{\alpha}{D^2} \left(\sum_{k=1}^M (\mathbb{X}'_k - \mathbb{X}'_{k-1}) \sqrt[3]{M - k + 1} \right)^3$, respectively. $\hat{\Phi}(\mathbb{X}) < \hat{\Phi}(\mathbb{X}')$ because

$$\begin{aligned}&\sum_{k=1}^M (\mathbb{X}_k - \mathbb{X}_{k-1}) \sqrt[3]{M - k + 1} - \sum_{k=1}^M (\mathbb{X}'_k - \mathbb{X}'_{k-1}) \sqrt[3]{M - k + 1} \\ &= \sum_{k=i, j} ((\mathbb{X}_k - \mathbb{X}'_k) \sqrt[3]{M - k + 1} - (\mathbb{X}_k - \mathbb{X}'_k) \sqrt[3]{M - k}) \\ &= (\mathbb{X}_i - \mathbb{X}'_i) [(\sqrt[3]{M - i + 1} - \sqrt[3]{M - i}) - (\sqrt[3]{M - j + 1} - \sqrt[3]{M - j})] \\ &< 0. \quad \square\end{aligned}$$

When $p_1 = 0$, Algorithm LTF always generates an optimal schedule because no core is associated with more than one task. *For the rest of this section, suppose that $p_1 \neq 0$.* We first derive an upper bound on $\Phi(\Psi^{LTF})$ for any schedule Ψ^{LTF} derived by Algorithm LTF and then a lower bound on the optimal energy consumption for T (regardless of which algorithm is adopted). Let $\hat{n}_k = |\{i \mid p_i \leq k \cdot p_1\}|$, and $P_k = \sum_{i=1}^{\hat{n}_k} p_i$ for some real $k \geq 1$. $\mathbb{X}^{LTF}(k) = \{\hat{p}_1, \hat{p}_2, \dots, \hat{p}_M\}$ is revised based on \mathbb{X}^{LTF} by load redistribution as follows:

1. $\hat{p}_i \leftarrow p_i$ if $i > \hat{n}_k$;
2. $\hat{p}_i \leftarrow p_1$ if $\lfloor \frac{k \cdot \hat{n}_k p_1 - P_k}{p_1(k-1)} \rfloor \geq i \geq 1$;
3. $\hat{p}_i \leftarrow k \cdot p_1$ if $\hat{n}_k \geq i \geq \lceil \frac{k \cdot \hat{n}_k p_1 - P_k}{p_1(k-1)} \rceil + 1$;
4. $\hat{p}_i \leftarrow P_k - p_1(\lceil \frac{k \cdot \hat{n}_k p_1 - P_k}{p_1(k-1)} \rceil - 1) - k \cdot p_1(\hat{n}_k - \lceil \frac{k \cdot \hat{n}_k p_1 - P_k}{p_1(k-1)} \rceil)$ if $i = \lceil \frac{k \cdot \hat{n}_k p_1 - P_k}{p_1(k-1)} \rceil$.

Lemma 4 *The minimum energy consumption to schedule tasks based on $\mathbb{X}^{LTF}(k)$ is no less than that based on \mathbb{X}^{LTF} for any real $k \geq 1$.*

Proof. Initially, let $\mathbb{X} = \mathbb{X}^{LTF}$. We could repeat the following revision procedure of \mathbb{X} until $i \geq j$ (where i and j are the smallest and the largest indices which satisfy $\mathbb{X}_i > p_1$ and

$\mathbb{X}_j < kp_1$, respectively): $\mathbb{X}_i \leftarrow \mathbb{X}_i - \delta$ and $\mathbb{X}_j \leftarrow \mathbb{X}_j + \delta$, where $\delta \leftarrow \min\{\mathbb{X}_i - p_1, kp_1 - \mathbb{X}_j\}$. The final load distribution \mathbb{X} would be as the same as $\mathbb{X}^{LTF}(k)$ after a finite number of the above procedure applied. Based on Lemma 3, $\hat{\Phi}(\mathbb{X}^{LTF}(k)) \geq \hat{\Phi}(\mathbb{X}^{LTF})$. \square

For the simplicity of representation, we denote $\frac{k \cdot \hat{m}_k p_1 - P_k}{p_1(k-1)}$ as χ . We have the following inequality:

$$\hat{\Phi}(\mathbb{X}^{LTF}(k)) \leq \frac{\alpha}{D^2} [\sqrt[3]{M} \hat{p}_1 + \sqrt[3]{M - \chi} (k-1) \hat{p}_1 + \sum_{i=\hat{m}_k+1}^M (\hat{p}_i - \hat{p}_{i-1}) \sqrt[3]{M - i + 1}]^3, \quad (4)$$

where the inequality comes from the following inequality: $\sqrt[3]{M - \lfloor \chi \rfloor} (\hat{p}_{\lfloor \chi \rfloor} - \hat{p}_1) + \sqrt[3]{M - \lceil \chi \rceil} (\hat{p}_{\hat{m}_k} - \hat{p}_{\lceil \chi \rceil}) \leq \sqrt[3]{M - \chi} (k-1) \hat{p}_1$.

Let $T' = \{\tau_i \mid c_i \geq \frac{\sum_{j=i+1}^{|T|} c_j}{M-i}\}$. We shall show that one core will be selected to service only one task in T' , regardless of whether X^{LTF} or an optimal task assignment X^{OPT} is considered. When a task $\tau_j \notin T'$ is considered in Algorithm LTF (i.e., Steps 3-5), there must exist a core m^* whose load p_{m^*} is no more than c_i for any $\tau_i \in T'$. Therefore, no other task will be assigned to any core occupied by any $\tau_i \in T'$ by Algorithm LTF. Consider any task assignment X derived by some algorithm. If some task $\tau_i \in T'$ and some other task τ_j are assigned on the same core where $c_i \geq c_j$, another task assignment X' could always be generated by moving τ_j to another core m' where $\mathbb{X}_{m'} < c_i$. The optimal X' -based schedule consumes less energy than that based on X . Thus X must not be the optimal task assignment. Therefore, one core will be selected to service only one task in T' for an optimal task assignment X^{OPT} . Let $\mathbb{X}^{OPT} = \{q_1, q_2, \dots, q_M\}$ be the load distribution for an optimal solution. Note that $\hat{m}_k = \lfloor i \mid p_i \leq k \cdot p_1 \rfloor$. We could prove the following lemma.

Lemma 5 *If $k = 2$, then $q_{\hat{m}_k+i} = p_{\hat{m}_k+i}$, for all $1 \leq i \leq M - \hat{m}_k$.*

Proof. Let j be the largest index for a core to which a task $\tau_n \in T - T'$ is assigned, and $\tau_n \in X_j^{LTF}$. Let the last task inserted into X_j^{LTF} be τ_r . Since $c_n < \frac{\sum_{j=|T'|+1}^{|T|} c_j}{M-|T'|}$, $|X_j^{LTF}| \geq 2$. It is clear that $c_r \leq p_1$ and $p_j - c_r \leq p_1$. Therefore, $p_j \leq 2p_1$. Since $\mathbb{X}_h^{LTF} > 2p_1$ for any $h > \hat{m}_2$, we have $j \leq \hat{m}_2$. Furthermore, we know that $q_{j+i} = p_{j+i}$, for all $1 \leq i \leq M - j$. \square

Note that $P_k = \sum_{i=1}^{\hat{m}_k} p_i$, where \hat{m}_k is defined by $\mathbb{X}^{LTF}(k)$. Similar to the definition of $\mathbb{X}^{LTF}(k)$, we define $\bar{\mathbb{X}}^{OPT}(k)$ as an adjusted load distribution according to \mathbb{X}^{OPT} by re-distributing $\sum_{i=1}^{\hat{m}_k} q_i$ so that $\hat{q}_1 = \hat{q}_2 = \dots = \hat{q}_{\hat{m}_k} = \frac{\sum_{i=1}^{\hat{m}_k} q_i}{\hat{m}_k}$. Similar to the proof in Lemma 4, we have $\hat{\Phi}(\mathbb{X}^{OPT}) \geq \hat{\Phi}(\bar{\mathbb{X}}^{OPT}(k))$ for any $k \geq 1$. We conclude this section by showing the following theorem.

Theorem 1 *Algorithm LTF has a 2.371-approximation ratio for the ECMS problem.*

Proof. The approximation ratio A^{LTF} is:

$$\begin{aligned} A^{LTF} &= \frac{\hat{\Phi}(\mathbb{X}^{LTF})}{\hat{\Phi}(\mathbb{X}^{OPT})} \leq \frac{\hat{\Phi}(\mathbb{X}^{LTF}(2))}{\hat{\Phi}(\bar{\mathbb{X}}^{OPT}(2))} \\ &\leq \frac{((\sqrt[3]{M} + \sqrt[3]{M - \chi}) \hat{p}_1 + \sum_{i=\hat{m}_2+1}^M (\hat{p}_i - \hat{p}_{i-1}) \sqrt[3]{M - i + 1})^3}{(\sqrt[3]{M} \hat{q}_1 + \sqrt[3]{M - \hat{m}_2} (2\hat{p}_1 - \hat{q}_1) + \sum_{i=\hat{m}_2+1}^M (\hat{p}_i - \hat{p}_{i-1}) \sqrt[3]{M - i + 1})^3} \\ &\leq \frac{((\sqrt[3]{M} + \sqrt[3]{M - \chi}) \hat{p}_1)^3}{(\sqrt[3]{M} \hat{q}_1 + \sqrt[3]{M - \hat{m}_2} (2\hat{p}_1 - \hat{q}_1))^3} \leq \left(\frac{\sqrt[3]{M} + \sqrt[3]{\frac{L - M \hat{p}_1}{\hat{p}_1}}}{\sqrt[3]{M} \frac{L}{M}} \right)^3, \quad (5) \end{aligned}$$

where L is $\hat{q}_1 \hat{m}_2 + 2\hat{p}_1(M - \hat{m}_2)$. Note that $\hat{q}_1 \hat{m}_2 = \sum_{i=1}^{\hat{m}_2} q_i = P_2$. We have $L = \chi \hat{p}_1 + 2\hat{p}_1(M - \chi)$. Let $f(x)$ be defined as $f(x) = \frac{(\sqrt[3]{M} + \sqrt[3]{\frac{K - Mx}{M}})x}{\sqrt[3]{M} \frac{K}{M}}$ for any rational number K where $x \cdot a + 2x \cdot (M - a) = K$ for some non-negative rational number a . By solving the equation $f'(x) = 0$, where $f''(x) < 0$, we have

$$f(x) \leq \frac{4}{3}, \quad (6)$$

where the maximal value stands when $x = \frac{8K}{9M}$. According to Equations (5) and (6), we have $A^{LTF} \leq \left(\frac{4}{3}\right)^3 < 2.371$. \square

5 Simulation Results

The purpose of this section is to provide performance evaluation of Algorithm LTF. Algorithm RAND was also simulated for reference, where the Algorithm RAND greedily assigned a task to any core with the minimum load without sorting tasks. The *relative energy consumption ratio*, which was defined as $\frac{\Phi(\Psi^{LTF})}{\Phi(\Psi^{OPT})}$, was adopted as the performance metric, where Ψ^{OPT} is an optimal schedule for the ECMS problem. Ψ^{OPT} can be obtained via an exhaustive search with a branch and bound strategy. When $|T|$ was a large number, the *relaxed relative energy consumption ratio*, which was defined as $\frac{\Phi(\bar{\mathbb{X}}^{LTF})}{\Phi(\bar{\mathbb{X}}^{OPT}(2))}$ (please refer to Section 4 for the definition of $\bar{\mathbb{X}}^{OPT}(k)$) was adopted as the performance metric. By definitions, $\hat{\Phi}(\bar{\mathbb{X}}^{OPT}(2))$ can be obtained in an efficient manner².

D was set as any arbitrary positive rational number in the simulations. The amount of cycles c_i for a task τ_i was generated randomly in the range $(0, D]$. The power consumption function $P(s)$ was s^3 . 100 independent simulations were run for each parameter configuration. When the results were for the average relative energy consumption ratio, their results were averaged. When they were for the maximum relative energy consumption ratio, the maximum value was returned. Figure 1(a) and (b) show the average and maximum relative energy consumption ratios for the simulated algorithms, when the number of cores ranged from 3 to 8, and the task set size ranged from 10 to 15. Figure 1(c) and (d) show the average and maximum relaxed relative energy consumption ratios for the simulated algorithms, when the number of cores ranged from 8 to 32, and the task set size ranged from 50 to 100. The

²Since the problem is NP-hard, the performance metric *relaxed relative energy consumption ratio* aimed at the providing of an approximate index when the optimal solution could not be obtained "efficiently".

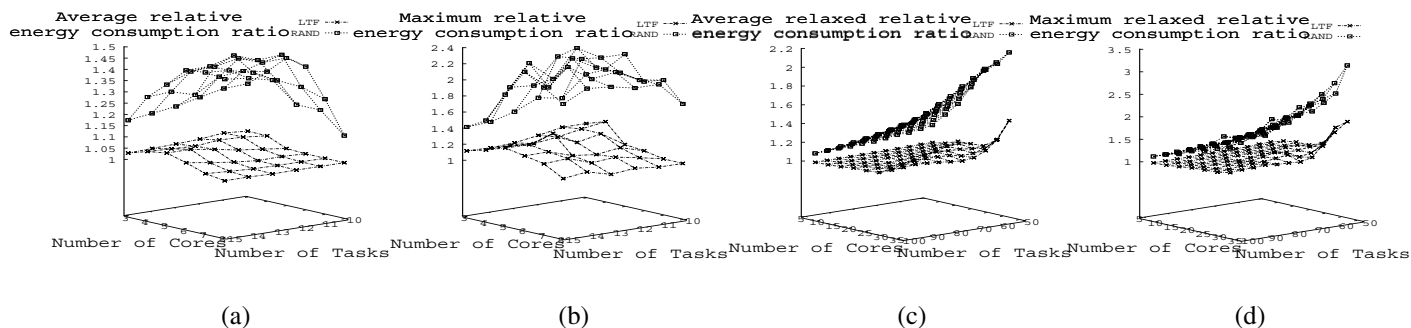


Figure 1. The simulation results of Algorithm LTF and Algorithm RAND: (a) The average relative energy consumption ratio when $|T| = 10 \dots 15$ and $M = 3 \dots 8$ (b) The maximum relative energy consumption ratio when $|T| = 10 \dots 15$ and $M = 3 \dots 8$ (c) The average relaxed relative energy consumption ratio when $|T| = 50 \dots 100$ and $M = 8 \dots 32$ (d) The maximum relaxed relative energy consumption ratio when $|T| = 50 \dots 100$ and $M = 8 \dots 32$

maximum and average relative energy consumption ratios for Algorithm LTF were less than 1.36 and 1.07 respectively. Furthermore, the maximum and average relaxed relative energy consumption ratios for Algorithm LTF were less than 2.00 and 1.44, respectively.

6 Conclusion

In this paper, we explore real-time energy-efficient scheduling on a chip multiprocessor with dynamic voltage scaling. We consider frame-based task sets, in which all tasks are ready at time 0 and share a common deadline. When a task partition is given, we present an optimal scheduling algorithm for the minimization of energy consumption. When task partitioning and scheduling must be resolved, we first prove the NP-hardness of the problem and then propose a 2.371-approximation algorithm with $O(|T|(\log |T| + \log M) + M)$, where T is a given task set, and M is the number of cores for a chip multiprocessor. A series of simulations was conducted the strength of our proposed algorithm, for which we have very encouraging results.

References

- [1] J. H. Anderson and S. K. Baruah. Energy-efficient synthesis of periodic task systems upon identical multiprocessor platforms. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, pages 428–435, 2004.
- [2] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 95–105, 2001.
- [3] A. Chandrakasan, S. Sheng, and R. Broderson. Lower-power CMOS digital design. *IEEE Journal of Solid-State Circuit*, 27(4):473–484, 1992.
- [4] J.-J. Chen, H.-R. Hsu, K.-H. Chuang, C.-L. Yang, A.-C. Pang, and T.-W. Kuo. Multiprocessor energy-efficient scheduling with task migration considerations. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, pages 101–108, 2004.
- [5] J.-J. Chen, T.-W. Kuo, and C.-L. Yang. Profit-driven uniprocessor scheduling with energy and timing constraints. In *ACM Symposium on Applied Computing*, pages 834–840. ACM Press, 2004.
- [6] J. Y. Chen, W. B. Jone, J. S. Wang, H.-I. Lu, and T. F. Chen. Segmented bus design for low-power systems. *IEEE Transactions on VLSI Systems*, 7(1):25–29, 1999.
- [7] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co, 1979.
- [8] F. Gruian. System-level design methods for low-energy architectures containing variable voltage processors. In *Power-Aware Computing Systems*, pages 1–12, 2000.
- [9] F. Gruian and K. Kuchcinski. Lenex: Task scheduling for low energy systems using variable supply voltage processors. In *Proc. Asia South Pacific Design Automation Conference*, pages 449–455, 2001.
- [10] V. Gutnik and A. P. Chandrakasan. Embedded power supply for low-power DSP. *IEEE Transactions on VLSI Systems*, 5(4):425–435, 1997.
- [11] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 37–46. Society for Industrial and Applied Mathematics, 2003.
- [12] T. Ishihara and H. Yasuura. Voltage scheduling problems for dynamically variable voltage processors. In *Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202, 1998.
- [13] W.-B. Jone, J. S. Wang, H.-I. Lu, I. P. Hsu, and J.-Y. Chen. Design theory and implementation for low-power segmented bus systems. *ACM Transactions on Design Automation of Electronic Systems*, 8(1):38–54, 2003.
- [14] P. Mejía-Alvarez, E. Levner, and D. Mossé. Adaptive scheduling server for power-aware real-time tasks. *ACM Transactions on Embedded Computing Systems*, 3(2):284–306, 2004.
- [15] R. Mishra, N. Rastogi, D. Zhu, D. Mosse, and R. Melhem. Energy aware scheduling for distributed real-time systems. In *International Parallel and Distributed Processing Symposium*, page 21, 2003.
- [16] M. Nikitovic and M. Brorsson. An adaptive chip-multiprocessor architecture for future mobile terminals. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 43–49, 2002.
- [17] M. Pedram and J. M. Rabaey. *Power Aware Design Methodologies*. Kluwer Academic Publishers, 2002.
- [18] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *Proceedings of Symposium on Operating Systems Design and Implementation*, pages 13–23, 1994.
- [19] F. Yao, A. Demers, and S. Shankar. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 374–382. IEEE, 1995.
- [20] H.-S. Yun and J. Kim. On energy-optimal voltage scheduling for fixed-priority hard real-time systems. *ACM Transactions on Embedded Computing Systems*, 2(3):393–430, Aug. 2003.
- [21] Y. Zhang, X. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *Annual ACM IEEE Design Automation Conference*, pages 183–188, 2002.
- [22] D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems. In *Proceedings of IEEE 22th Real-Time System Symposium*, pages 84–94, 2001.