Applied Mathematical Modelling 38 (2014) 3456–3466

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Energy efficient voltage scheduling for multi-core processors with software controlled dynamic voltage scaling

Abhishek Mishra [a,*], Anil Kumar Tripathi [b]

[a] Center of Excellence in Information and Communication Technology, Indian Institute of Technology Jodhpur, Old Residency Road, Ratanada, Jodhpur 342 011, India
[b] Department of Computer Engineering, Indian Institute of Technology (Banaras Hindu University), Varanasi 221 005, India

ARTICLE INFO

Article history:
Received 21 January 2013
Received in revised form 24 September 2013
Accepted 18 December 2013
Available online 4 January 2014

Keywords:
Dynamic voltage scaling
Energy efficient scheduling
Integer linear programming
Multi-core processors

ABSTRACT

Energy efficient voltage scheduling for multi-core processors is an important issue in the context of parallel and distributed computing. Dynamic voltage scaling (DVS) is used to reduce the energy consumption of cores. Nowadays processor vendors are providing software for DVS. We consider a system using a single multi-core processor with software controlled DVS having a finite set of discretely available core speeds. Our contribution to this work is solving a well-known energy efficient voltage scheduling problem on the considered system. The problem that we consider is to find a minimum energy voltage scheduling for a given computational load that has to be completed within a given deadline. First we show that the existing methods to solve this problem on other processor models fail to apply on our processor model. Then we formulate an Integer Program (IP) for the problem. Through a series of reductions we reduce the IP formulation of the problem into an Integer Linear Program (ILP) formulation and prove that the proposed IP for the problem can be solved in $O(D(log(max(s_{max}, p) + 1) + qlog(Dp + 1)) + log(\alpha p s_{max}^3 D)(2q(4q + 3)log(max(Dp, C) + 2))^{a^{2q}})$ time where D is the given deadline, C is the amount of computation that has to be completed within the deadline of D time units, p is the number of cores, q is the number of possible core speeds, s_{max} is the maximum speed of cores, and α and a are constants.

1. Introduction

Multi-core processor architectures are designed to meet the challenges of high performance computing applications [1,2]. A multi-core processor has more than one core on a single chip. This greatly reduces the hardware size. This also greatly reduces the communication delay between cores as compared to multiprocessors. The result is that we have improved computational performance with reduced hardware.

Dynamic voltage scaling (DVS) is a technique in which varying the supply voltage can vary the speed of cores [3]. This results in saving of energy when computational load is low.

This gives rise to the voltage scheduling problem [4] for multi-core processors in which the objective is to find a schedule of core speeds/voltages so as to minimize the energy consumption given a computational load to process and a deadline to meet.

We consider a system using a single multi-core processor with software controlled DVS having a finite set of discretely available core speeds. We consider the problem of finding a minimum energy voltage scheduling for a given computational load that has to be completed within a given deadline on the proposed processor model. We prove that this problem can be

* Corresponding author. Tel.: +91 291 244 9032.
E-mail addresses: amishra@iitj.ac.in (A. Mishra), aktripathi.cse@itbhu.ac.in (A.K. Tripathi).

0307-904X/$ - see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.apm.2013.12.009

solved in $O(D(log(max(s_{max},p)+1)+qlog(Dp+1))+log(\alpha p s_{max}^3 D)(2q(4q+3)log(max(Dp,C)+2))^{a^{2q}})$ time where D is the given deadline, C is the amount of computation that has to be completed within the deadline of D time units, p is the number of cores, q is the number of possible core speeds, s_{max} is the maximum speed of cores, and α and a are constants.

The rest of the paper is organized as follows: Section 2 highlights some of the previous work from the literature that is related to our work. In Section 3 we consider a system using a single multi-core processor with software controlled DVS having a finite set of discretely available core speeds. In Section 4 we consider the *Energy Efficient Load Scheduling Problem (EELSP)* on the proposed processor model. In Section 5 we give an $O(D(log(max(s_{max},p)+1)+qlog(Dp+1))+log(\alpha p s_{max}^3 D)(2q(4q+3)log(max(Dp,C)+2))^{a^{2q}})$ time algorithm for solving the *EELSP* problem. In Section 6 we give some experimental results. And finally we conclude in Section 7.

2. Related work

Energy efficient scheduling is an active area of research. There are many uniprocessor energy efficient scheduling algorithms proposed in the literature [5–8,4,9–12].

Ishihara and Yasuura [4] considered the problem of voltage scheduling on a uniprocessor with DVS that can use only a small number of discretely variable voltages. They proved that the voltage scheduling problem of minimizing the energy consumption by a processor for processing a given computational load with a given deadline can be solved optimally in polynomial time with at most two voltages.

Chen et al. [6] considered the problem of profit-driven uniprocessor scheduling with energy and timing constraints. They considered both type of processor models: processors with a finite number of discrete processor speeds, and processors with an infinite number of continuous processor speeds. Each task has associated with it an amount of computation, and a profit if it is completed within a common deadline for all tasks. Energy constraint is given. A lower bound for system profit is also given. The problem is to select a subset of tasks that satisfies the energy, profit and deadline constraints while maximizing the system profit. They proposed a 2-approximation algorithm and a fully polynomial time approximation scheme for the problem. They also solved the previous problem [4] for a different energy consumption function that is a convex function.

Yao et al. [10] considered the problem of minimum energy scheduling of independent jobs with arrival times, deadlines, and a given amount of computation on a uniprocessor with variable speeds under the assumption that the power function is a convex function of the processor speed. They considered the case of discretely available processor speeds. They gave an $O(nlog^2(n)))$ time optimal offline algorithm for the problem where n is the number of jobs. They also proposed some heuristics for the online version of the problem.

Irani et al. [8] extended the previous problem [10] to include the case in which a processor can go into a sleep state. In the sleep state, the processor speed and its power consumption is 0, but a constant amount of energy is required to bring back the processor into a non-sleep mode. They proposed a 3-approximation algorithm for the offline version of the problem. They also proposed an online algorithm for the problem.

Chen et al. [7] extended the problem of Yao et al. [10] for the case of jobs with precedence constraints. They considered the case of weakly dynamic voltage scheduling in which speed change is not allowed in the middle of processing a job. They gave fully polynomial-time approximation schemes for two special cases of the problem and also proved the problem to be NP-Complete.

Yang et al. [12] considered the problem of energy efficient real time task scheduling with temperature dependent leakage [13,14] on a processor. A workload of C time units is given that must be done in every consecutive D time units. The problem that they solve is to schedule the given workload with the minimal total energy consumption and without violation of any timing constraint. They propose a pattern based approach in which they divide the given time horizon into several time segments with the same length. In each time segment the processor is in the active mode at the beginning for a fixed amount of time, and it is in the dormant mode at the end for the remaining amount of time. In the active mode the computation is advanced, while the dormant mode is used to reduce the temperature by cooling as well as by leakage power consumption.

There are also various multiprocessor energy efficient scheduling algorithms proposed in the literature [15–22].

Yang et al. [22] considered the problem of energy consumption minimization for a chip-multiprocessor with DVS that can use continuously varying processor speeds with no upper bound. The power consumed by the processor was assumed to be proportional to the cube of the processor speed. They considered the problem of voltage scheduling for a set of independent tasks with a common deadline. They gave an 2.371-approximation algorithm for the problem.

Zhang et al. [20] considered the problem of energy efficient scheduling of real time dependent tasks on a given number of variable voltage processors. They provided a two-phase framework to solve the problem. The first phase is the task assignment and ordering. The second phase is the voltage selection (VS) phase. The VS problem is formulated as an integer-programming (IP) problem. They proved that the IP problem for VS could be solved in polynomial time for the case of processors with continuous voltages.

Shieh and Pong [23] consider the energy efficient task scheduling problem for a multicore processor by taking into account the overheads in transition between different voltage levels. They give an ILP formulation for the problem. They have also proposed an online heuristic algorithm for the problem, and compared with the optimal offline algorithm.

Zhuravlev et al. [24] survey various types of energy efficient task scheduling algorithms for multicore systems. They consider proactive thermal management techniques, in which temperature of cores are kept within a safe limit. They consider three types of thermal management techniques: static management technique, in which task scheduling is done at the start.

Nonpredictive proactive thermal management: in which scheduling is done at run time, it is non-predictive because there is no prediction of core temperatures. Predictive proactive thermal management: in which scheduling is done at run time, and also core temperatures are predicted. They have also surveyed single-threaded and multithreaded applications on assymetric multicore systems having different non-identical cores.

Chen et al. [25] consider the problem of dynamic voltage and frequency scaling for shared resources in multicore processors. They propose a method for dynamic voltage/frequency scaling of networks-on-chip and last level caches in multicore processor designs, where the shared resources form a single voltage/frequency domain. They have developed techniques for monitoring and control, and validated through system simulations on the PARSEC benchmarks.

Min-Allah et al. [26] consider power efficient rate monotonic scheduling for multi-core systems. They first try to find the lowest core speed so that deadline is satisfied, and energy consumption is low. They then shift the lightest tasks to different cores to maximize the core utilization.

Etinski et al. [27] propose a model that gives an upper bound on loss in performance due to frequency scaling using the application parallel efficiency. They have also validated their model using performance measurements of large scale parallel applications.

Zhu et al. [28] consider the problem of energy efficient task scheduling on a heterogeneous cluster with DVS. They propose an adaptive energy-efficient scheduling (AEES) for aperiodic and independent real-time tasks on heterogeneous clusters with DVS. The AEES algorithm adjusts the voltages according to the workload conditions of a cluster. When the cluster is heavily loaded, the AEES algorithm considers voltage levels of the new tasks as well as the tasks running currently to meet the deadlines. When the cluster is lightly loaded, the AEES algorithm reduces the voltage levels to conserve energy while also satisfying the deadlines.

Lee [29] has considered the problem of energy efficient scheduling of periodic realtime tasks on lightly loaded multicore processors. A multicore processor is said to be lightly loaded if the number of cores is greater than the number of tasks. A task is executed in parallel with a low speed on an idle core. The problem is proved to be NP-hard on a lightly loaded multicore processor. Then a polynomial-time scheduling scheme is proposed.

Dhiman et al. [30] have proposed a multi-tiered software system for energy efficient computing in virtualized environments called vGreen. It consists of metrics that capture power and performance characteristics of virtual and physical machines, and use it for energy efficient virtual machine scheduling.

Bergamaschi et al. [31] have considered an integrated power management (PM) unit in a multi-core processor, which monitors the performance and power of each core and dynamically changes the individual voltages and frequencies to maximize the system performance under a given power budget. They have performed experiments for both types of DVS: *per-chip DVS* and *per-core DVS*.

3. Systems using a single multi-core processor

3.1. Multi-core processors with software controlled DVS

Some examples of software controlled DVS are *Enhanced Intel SpeedStep Technology* [32], and *AMD PowerNow! Technology* [33]. Our algorithm and problem formulation technique is practical for multi-core processors that have a small number of cores with software controlled DVS having a small number of discretely available core speeds. For example: *Enhanced Intel SpeedStep Technology* [32] for the Intel Pentium M processor supports processor speeds of 600 MHz to 1.6 GHz with a step of 200 MHz. *AMD PowerNow! Technology* [33] supports the complete frequency operating range of the processor in use allowing steps of 33 or 50 MHz from an absolute low of 133 or 200 MHz. Our motivation for using a *single multi-core processor* with *per-chip DVS* comes from the *Intel (R) microarchitecture (Nehalem)* [34]. In the Intel (R) Xeon (R) processor series 3500 and 5500 using the Intel (R) microarchitecture (Nehalem) [34] there are four cores. Each core runs at the same speed. Each core can independently go to the sleep state. Depending upon the number of active cores, estimated current consumption, estimated power consumption, and the processor temperature, the cores can increase one (133.33 MHz) or two (266.66 MHz) frequency steps.

3.2. Notation

Let $\mathcal{N}$ denote the set of natural numbers. For a rational number x, $\lceil x \rceil$ is the smallest integer greater than or equal to x. In this work we are only dealing with integer vectors having each component as an integer. A d-dimensional vector V is written as $V = (v_i)_d$ where i is used to index the components of V as v_i where $i \in [1, d] \cap \mathcal{N}$. For two d-dimensional vectors U and V, their dot product is denoted as $U * V = \sum_{i=1}^{d} u_i v_i$. For an integer n we denote the vector V^n as $V^n = (v_i^n)_d$ where $V = (v_i)_d$. $min()$ and $max()$ are functions used to return the minimum and maximum of the input parameters respectively.

3.3. Characteristics of the considered multi-core processor

We take the power consumption function of the multi-core processor same as in Chandrakasan et al. [35], Weiser et al. [36], and Yang et al. [22]:

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

where

$$s = k_h((V_{dd} - V_t)^2)/V_{dd}. \tag{2}$$

Here $P()$ is the power consumption function, s is the processor speed, C_{ef} is the effective capacitance, V_t is the threshold voltage, V_{dd} is the supply voltage and k_h is a hardware dependent constant. We assume that:

$$V_{dd} \gg V_t. \tag{3}$$

From (1)–(3) we get:

$$P(s) = \alpha s^3, \tag{4}$$

where α is a constant.

Let the energy consumed by a core running at a speed of s for τ time units be given by $P(s)\tau$. Let the overheads in changing the supply voltages be negligible. We are assuming that any core can be taken into a sleep mode with $s = 0$, but all of non-sleeping cores must run at the same speed [22]. Let the computational work done c in cycles of a core running at a speed of s for τ time units be given by:

$$c = s\tau. \tag{5}$$

Our considered multi-core processor has software controlled DVS. Too frequent speed/voltage switching may cause unnecessary overhead on the system. To get a fine-grained control over power management we can set periodic checkpoints of time period $\delta\tau$ at which the DVS software checks whether to switch the speed/voltage or not. For example the *Crusoe Long-Run Power Management* [37] for the Crusoe processor supports up to 200 speed/voltage changes per second. Azevedo et al. [38] propose a profile-based dynamic voltage scheduling heuristic using program checkpoints. Program checkpoints are generated at compile time and indicate places in the code where the core speed/voltage should be recalculated. One advantage of using periodic checkpoints over program checkpoints is that the periodic checkpoints can be set at the run time.

For having low overhead in switching the speed/voltage, a good idea is to implement the DVS software as a periodic process that wakes up periodically to check if there is a need to change the voltage, and otherwise it sleeps. This means that after finishing its computation a core cannot go into the sleep mode abruptly. It has to wait for the next periodic invocation of the DVS software. For the remaining period of time it will remain idle wasting the energy. When we use software based DVS, the execution time will be slightly more as compared to the hardware based DVS. This means that the software based DVS will consume slightly more energy as compared to the hardware based DVS for the same speed/voltage schedule. The advantage of software based DVS over hardware based DVS is that in software based DVS, we can have full control in the way we schedule the speed/voltage, which is not possible in hardware based DVS.

In our multi-core processor model there are p homogeneous cores. We are assuming that the DVS software is a periodic process so that the supply voltage can change only in steps of a certain amount of time ($= \delta\tau$). Without loss of generality we can take this time step as our unit of time ($\delta\tau = 1$). There are q possible core speeds (non-zero) that are given by the set Q:

$$Q = \{s_i | (i \in [1, q] \cap \mathcal{N}) \wedge (s_i \in \mathcal{N})\}. \tag{6}$$

Let s_{max} be the maximum element of Q.

4. The *Energy Efficient Load Scheduling Problem (EELSP)*

4.1. Problem statement and IP formulation

We are given a computational work of C ($C \in \mathcal{N}$) cycles of a core that is to be finished within a deadline of D ($D \in \mathcal{N}$) time units. The problem is to find an optimal scheduling of voltages (or equivalently an optimal scheduling of speeds) that minimizes the energy consumption of the multi-core processor.

Let the core processor speeds scheduled during the time interval $[0, D)$ be the vector X given by:

$$X = (x_j)_D, \tag{7}$$

where x_j is the speed of non-sleeping cores during the time interval $[j-1, j)$. For $j \in [1, D] \cap \mathcal{N}$, we have the following constraints for x_j:

$$x_j \in \{0\} \cup Q. \tag{8}$$

Let the number of non-sleeping cores during the time interval $[j-1, j)$ be y_j, and let Y be the vector given by:

$$Y = (y_j)_D. \tag{9}$$

For $j \in [1, D] \cap \mathcal{N}$, we have the following constraints for y_j:

$$0 \leqslant y_j \leqslant p. \tag{10}$$

The energy function $E(X, Y)$ is given by:

$$E(X, Y) = \alpha X^3 * Y. \tag{11}$$

We also have the work constraint given by:

$$X * Y \geqslant C. \tag{12}$$

Definition 1. Given the input (C, D), the *Energy Efficient Load Scheduling Problem (EELSP)* is to find D-dimensional vectors X and Y such that (11) is minimized while also satisfying the constraints (8), (10), and (12).

4.2. Existing approaches

Ishihara and Yasuura [4] solved this problem for the case of a uniprocessor with DVS that can use only a small number of discretely variable voltages. They proved that the problem can be solved optimally in polynomial time with at most two voltages. Their approach can be summarized as follows: first they solve this problem for the case of uniprocessor with continuously available voltages from 0 to V_{max}. They proved that the minimum energy is obtained when the processor runs at a constant voltage V_{avg}, so that it completes the computation just at the deadline time D. For the case of uniprocessor with DVS that can use only a small number of discretely variable voltages, their result states that the minimum energy can be obtained with at most two voltages V_1 and V_2 that are just adjacent to V_{avg}:

$$V_1 < V_{avg} < V_2. \tag{13}$$

The ratio of time intervals at which the processor runs at the voltages V_1 and V_2 can easily be computed such that the computation just finishes on the deadline time D.

The processor model considered by Ishihara and Yasuura [4] used continuous time in the sense that the voltage changes can occur at any time. Our multi-core processor model uses discrete time in the sense that the speed changes can occur only at the end of the discrete time periods due to the periodic invocation of the DVS software. If we remove the restriction of periodic DVS software, and make the time continuous, then the results of Ishihara and Yasuura [4] can easily be extended to the case of multi-core processors. But their result cannot be extended to our model that uses discrete time. A simple example will illustrate this point: consider the problem instance $(\alpha = 1, p = 1, q = 3, Q = \{1, 10, 100\}, C = 111, D = 3)$. Applying the method of Ishihara and Yasuura [4] to our model, first we calculate the speed s_{avg} at which the processor just finishes the computation on deadline:

$$s_{avg} = C/D = 111/3 = 37. \tag{14}$$

s_{avg} immediately lies between $s_2 = 10$, and $s_3 = 100$. The only feasible solutions using the two speeds s_2 and s_3 are given by:

$$X = (10, 10, 100), \quad \text{and} \quad Y = (1, 1, 1), \quad \text{with} \quad E(X, Y) = 1002000, \tag{15}$$

and

$$X = (10, 100, 100), \quad \text{and} \quad Y = (1, 1, 1), \quad \text{with} \quad E(X, Y) = 2001000. \tag{16}$$

Now consider the following solution:

$$X = (1, 10, 100), \quad \text{and} \quad Y = (1, 1, 1), \quad \text{with} \quad E(X, Y) = 1001001. \tag{17}$$

It is clear that the solution (17) gives a solution with lower energy consumed as compared to solutions (15) and (16). The reason for non-applicability of the approach of Ishihara and Yasuura [4] to our model is the wastage of energy for fractional allocations. When a core becomes idle in between a time period, for the remaining period of time it wastes the energy because it can go to the sleep state only on the next periodic invocation of the DVS software. This wastage of energy makes the problem difficult.

5. An $O(D(\log(\max(s_{max}, p)) + 1) + q\log(Dp + 1)) + \log(\alpha p s_{max}^3 D)(2q(4q + 3)\log(\max(Dp, C) + 2))^{a^{2q}})$ time algorithm for solving the *EELSP* problem

General ILP problem is NP-Complete [39]. The difficulty with the *EELSP* problem is the number of variables. X and Y are D-dimensional vectors. The dimension of the vectors is itself a variable. In the *EELSP* problem we consider the speed that is allocated to each time period. In an attempt to simplify the problem, we use a slightly different formulation of the problem. For each speed we consider the number of time periods for all the cores that are assigned that speed. For this purpose we introduce a new vector W. Let W be a q-dimensional vector given by:

$$W = (w_i)_q. \tag{18}$$

Let S be the q-dimensional vector given by:

$$S = (s_i)_q. \tag{19}$$

Let the energy function $E(W)$ be defined as:

$$E(W) = \alpha S^3 * W. \tag{20}$$

Let the work constraint be defined as:

$$S * W \geqslant C. \tag{21}$$

Let there be the following constraints for $i \in [1, q] \cap \mathcal{N}$:

$$0 \leqslant w_i \leqslant Dp, \tag{22}$$

$$\sum_{i=1}^{q} w_i \leqslant Dp \tag{23}$$

and

$$\sum_{i=1}^{q} \lceil w_i/p \rceil \leqslant D. \tag{24}$$

Constraint (22) states that at most we can run each core for each time period with a single speed within the deadline. Constraint (23) states that at most we can run each core for each time period with any speed within the deadline. For any feasible solution, one of the speed allocations with the minimum completion time is such that at most one time period for each speed can have sleeping cores. Constraint (24) states that such an allocation should be completed within the deadline.

Definition 2. Given the input (C, D), the *EELSP2* problem is to find a q-dimensional vector W such that (20) is minimized while also satisfying the constraints (21)–(24).

Our next step is proving that the *EELSP2* problem can be used to solve the *EELSP* problem.

Proposition 1. *Given a W minimizing EELSP2, in* $O(D\log(\max(s_{max}, p) + 1))$ *time we can find X and Y minimizing EELSP.*

Proof. We construct X and Y from W such that at most one time period for each speed can have sleeping cores. Assuming that W minimizes *EELSP2*. For $j \in \left[\sum_{k=1}^{i-1} \lceil w_k/p \rceil + 1, \sum_{k=1}^{i} \lceil w_k/p \rceil\right] \cap \mathcal{N}$ let:

$$x_j = s_i. \tag{25}$$

For $j \in \left[\sum_{k=1}^{q} \lceil w_k/p \rceil + 1, D\right] \cap \mathcal{N}$ let:

$$x_j = 0. \tag{26}$$

Let X be the vector given by:

$$X = (x_j)_D. \tag{27}$$

For $j \in \left[\sum_{k=1}^{i-1} \lceil w_k/p \rceil + 1, \quad \sum_{k=1}^{i} \lfloor w_k/p \rfloor\right] \cap \mathcal{N}$ let:

$$y_j = p. \tag{28}$$

If $\lceil w_i/p \rceil - \lfloor w_i/p \rfloor = 1$, then for $j = \sum_{k=1}^{i} \lceil w_k/p \rceil$ let:

$$y_j = w_i - p\lfloor w_i/p \rfloor. \tag{29}$$

For $j \in \left[\sum_{k=1}^{q} \lceil w_k/p \rceil + 1, D\right] \cap \mathcal{N}$ let:

$$y_j = 0. \tag{30}$$

Let Y be the vector given by:

$$Y = (y_j)_D. \tag{31}$$

From (25)–(27) it follows that X satisfies the constraint (8). From (28)–(31) it follows that Y satisfies the constraint (10). From (25)–(31) it follows that:

$$S * W = X * Y, \tag{32}$$

and

$$E(W) = E(X, Y). \tag{33}$$

Therefore (12) follows from (21) and (32). This implies that X and Y are also satisfying the constraints of *EELSP*. If X and Y are not minimizing *EELSP*, then let X_1 and Y_1 minimize *EELSP*. Therefore we have:

$$E(X_1, Y_1) < E(X, Y). \tag{34}$$

Let W_1 be constructed from X_1 and Y_1 using the construction given in Proposition 2 (Eqs. (37) and (38)). From the construction given in Proposition 2, it follows that (Eq. (41)):

$$E(W_1) = E(X_1, Y_1). \tag{35}$$

From (33)–(35) it follows that:

$$E(W_1) < E(W), \tag{36}$$

contradicting our assumption that W minimizes *EELSP2*. Therefore X and Y minimize *EELSP*. Time taken in this conversion is $O(Dlog(max(s_{max}, p) + 1))$ because of (25)–(31). □

Proposition 2. *Given X and Y minimizing EELSP, in $O(qDlog(Dp + 1))$ time we can find a W minimizing EELSP2.*

Proof. Assuming that X and Y minimize *EELSP*. Let:

$$w_i = \sum_{j=1, x_j = s_i}^{D} y_j. \tag{37}$$

Let W be the vector given by:

$$W = (w_i)_q. \tag{38}$$

The constraints (22) and (23) follow from (10) and (37). From (10) and (37) we have:

$$\sum_{i=1}^{q} \lceil w_i/p \rceil = \sum_{i=1}^{q} \lceil \sum_{j=1, x_j = s_i}^{D} y_j/p \rceil \leqslant \sum_{i=1}^{q} \sum_{j=1, x_j = s_i}^{D} 1 = D. \tag{39}$$

Therefore (24) also holds. From (37) and (38) it follows that:

$$S * W = X * Y, \tag{40}$$

and

$$E(W) = E(X, Y). \tag{41}$$

Therefore (21) follows from (12) and (40). This implies that W is also satisfying the constraints of *EELSP2*. If W is not minimizing *EELSP2*, then let W_2 minimize *EELSP2*. Therefore we have:

$$E(W_2) < E(W). \tag{42}$$

Let X_2 and Y_2 be constructed from W_2 using the construction given in Proposition 1 (Eq. (25)–(31)). From the construction given in Proposition 1, it follows that (Eq. (33)):

$$E(X_2, Y_2) = E(W_2). \tag{43}$$

From (41)–(43) it follows that:

$$E(X_2, Y_2) < E(X, Y), \tag{44}$$

contradicting our assumption that X and Y minimize *EELSP*. Therefore W minimizes *EELSP2*. Time taken in this conversion is $O(qDlog(Dp + 1))$ because of (37) and (38). □

Let Z be a q-dimensional vector given by:

$$Z = (z_i)_q. \tag{45}$$

Let there be the following constraints for $i \in [1, q] \cap \mathcal{N}$:

$$p(z_i - 1) < w_i \leqslant pz_i \tag{46}$$

and

$$\sum_{i=1}^{q} z_i \leqslant D. \tag{47}$$

For removing the ceiling function in the constraint (24) of the *EELSP2* problem, we introduce a new vector Z. Constraint (46) is for enforcing z_i to be $\lceil w_i/p \rceil$. Constraint (47) is the same as the constraint (24) together with the constraint (46).

Definition 3. Given the input (C, D), the *EELSP3* problem is to find a q-dimensional vector W such that (20) is minimized while also satisfying the constraints 21, 22, 46, 47.

Proposition 3. *A W minimizing EELSP3 also minimizes EELSP2.*

Proof. Assuming that W minimizes *EELSP3*. Therefore the constraints (21) and (22) of *EELSP2* are satisfied by W. The constraints (23) and (24) of *EELSP2* follow from (46) and (47). This implies that W is also satisfying the constraints of *EELSP2*. If W is not minimizing *EELSP2*, then let W_3 minimize *EELSP2*. Therefore we have:

$$E(W_3) < E(W). \tag{48}$$

Using the construction of Proposition 4 (Eq. (49)), W_3 also satisfies *EELSP3*. The inequality (48) contradicts our assumption that W minimizes *EELSP3*. Therefore it follows that W also minimizes *EELSP2*. □

Proposition 4. *A W minimizing EELSP2 also minimizes EELSP3.*

Proof. Assuming that W minimizes *EELSP2*. Therefore the constraints (21) and (22) of *EELSP3* are satisfied. For $i \in [1,q] \cap \mathcal{N}$, let:

$$z_i = \lceil w_i/p \rceil. \tag{49}$$

From (49) we get (46). From (24) and (49), we get (47). This implies that W is also satisfying the constraints of *EELSP3*. If W is not minimizing *EELSP3*, then let W_4 minimize *EELSP3*. Therefore we have:

$$E(W_4) < E(W). \tag{50}$$

Using the construction of Proposition 3 (the identity mapping: $W \to W$), W_4 also satisfies *EELSP2*. The inequality (50) contradicts our assumption that W minimizes *EELSP2*. Therefore it follows that W also minimizes *EELSP3*. □

Theorem 5. *EELSP can be solved in* $O(D(log(max(s_{max},p)+1)+qlog(Dp+1))+log(\alpha ps_{max}^3 D)(2q(4q+3)log(max(Dp,C)+2))^{a^{2q}})$ *time.*

Proof. For an integer L, if we write the energy constraint as:

$$E(W) \leqslant L, \tag{51}$$

then *EELSP3* is an Integer Linear Program (ILP) with the number of variables and the number of constraints as system dependent constants (depending upon q). This can be solved by using Lenstra's algorithm [40] in time $O((2q(4q+3)log(max(Dp,C)+2))^{a^{2q}})$ where a is a constant. Minimum solution can be found by performing binary search [41] on L. Therefore the number of ILP's to be solved is bounded by $O(log(E_{max}))$ where E_{max} is the maximum possible energy consumed by the system during the time interval $[0,D)$ and is given by:

$$E_{max} = \alpha ps_{max}^3 D. \tag{52}$$

Using (52), and Proposition 1–4, we can solve *EELSP* in time $O(D(log(max(s_{max},p)+1)+qlog(Dp+1))+log(\alpha ps_{max}^3 D)(2q(4q+3)log(max(Dp,C)+2))^{a^{2q}})$. □

EELSA (Energy Efficient Load Scheduling Algorithm) is the desired algorithm for solving the *EELSP* problem.

Algorithm 1. ENERGY-EFFICIENT-LOAD-SCHEDULING-ALGORITHM (C,D)

```
01 L ← E_max
02 solve the ILP with constraints (51), (21), (22), (46), and (47) using Lenstra's algorithm [40]
03 if step 02 gives no solution
04   then return NO SOLUTION
05   else low ← 1
06     high ← L
07     while low < high
08       do mid ← ⌊(low + high)/2⌋
09         L ← mid
10         solve the ILP with constraints (51), (21), (22), (46), and (47) using Lenstra's algorithm [40]
11         if step 10 gives no solution
12           then low ← max(mid, low + 1)
13           else high ← min(mid, high − 1)
14 use the construction given in Proposition 1 to get X and Y from W
15 return (X, Y)
```

The EELSA algorithm can be applied for fine-grained tasks having low computational requirements. For coarse-grained tasks having high computational requirements, first we have to divide the computational load into tasks, and then design an energy efficient task scheduling algorithm.

6. Experimental results

We have compared the EELSA algorithm with the algorithm of Ishihara and Yasuura [4] (IYA). We have considered two types of processors: a unicore processor with $p = 1$, and a dual-core processor with $p = 2$. For each processor type we have taken four possible speed sets $Q : \{1, 2, 4\}, \{1, 4, 8\}, \{1, 8, 16\}$, and $\{5, 7, 11\}$. For each processor type p and speed set Q, we have taken five values of deadline $D : 3, 4, 5, 6$, and 7. We have selected the amount of computation required C ranging from 7 to 57 for unicore processors, and 13 to 113 for dual-core processors. Table 1 shows the result for unicore processors, and Table 2 shows the result for dual-core processors. E_{IYA} is the energy consumed by the IYA algorithm, and E_{EELSA} is the energy consumed by the EELSA algorithm. δE is the percentage improvement of the EELSA algorithm over the IYA algorithm, and is defined as:

Table 1
Comparison of IYA and EELSA for unicore processors.

Q	C	D	E_{IYA}	E_{EELSA}	δE (%)
$\{1,2,4\}$	7	3	80	73	9.59
$\{1,2,4\}$	9	4	88	81	8.64
$\{1,2,4\}$	11	5	96	89	7.87
$\{1,2,4\}$	13	6	104	97	7.22
$\{1,2,4\}$	15	7	112	105	6.67
$\{1,4,8\}$	13	3	640	577	10.92
$\{1,4,8\}$	17	4	704	641	9.83
$\{1,4,8\}$	21	5	768	705	8.94
$\{1,4,8\}$	25	6	832	769	8.19
$\{1,4,8\}$	29	7	896	833	7.56
$\{1,8,16\}$	25	3	5120	4609	11.09
$\{1,8,16\}$	33	4	5632	5121	9.98
$\{1,8,16\}$	41	5	6144	5633	9.07
$\{1,8,16\}$	49	6	6656	6145	8.32
$\{1,8,16\}$	57	7	7168	6657	7.68
$\{5,7,11\}$	23	3	2017	1799	12.12
$\{5,7,11\}$	30	4	2360	2142	10.18
$\{5,7,11\}$	45	5	4679	4461	4.89
$\{5,7,11\}$	44	6	3046	2828	7.71
$\{5,7,11\}$	51	7	3389	3171	6.87

Table 2
Comparison of IYA and EELSA for dualcore processors.

Q	C	D	E_{IYA}	E_{EELSA}	δE (%)
$\{1,2,4\}$	13	3	152	145	4.83
$\{1,2,4\}$	17	4	168	161	4.35
$\{1,2,4\}$	21	5	184	177	3.95
$\{1,2,4\}$	25	6	200	193	3.63
$\{1,2,4\}$	29	7	216	209	3.35
$\{1,4,8\}$	25	3	1216	1153	5.46
$\{1,4,8\}$	33	4	1344	1281	4.92
$\{1,4,8\}$	41	5	1472	1409	4.47
$\{1,4,8\}$	49	6	1600	1537	4.10
$\{1,4,8\}$	57	7	1728	1665	3.78
$\{1,8,16\}$	49	3	9728	9217	5.54
$\{1,8,16\}$	65	4	10752	10241	4.99
$\{1,8,16\}$	81	5	11776	11265	4.54
$\{1,8,16\}$	97	6	12800	12289	4.16
$\{1,8,16\}$	113	7	13824	13313	3.84
$\{5,7,11\}$	44	3	4034	3598	12.12
$\{5,7,11\}$	58	4	4720	4284	10.18
$\{5,7,11\}$	88	5	9358	8922	4.89
$\{5,7,11\}$	86	6	6092	5656	7.71
$\{5,7,11\}$	100	7	6778	6342	6.87

$$\delta E = 100(E_{IYA} - E_{EELSA})/E_{EELSA}. \quad (53)$$

As is evident from Tables 1 and 2, using the EELSA algorithm, we can get up to 12.12% improvement over the IYA algorithm.

7. Conclusion

We proposed a model of multi-core processors with software controlled DVS that has a finite set of discretely available core speeds. We studied the problem of voltage scheduling to minimize the energy consumption given a computational load to process and a deadline to meet. After showing the ineffectiveness of existing methods, we formulated the problem as an ILP (the *EELSP* problem) and proved that the problem can be solved in $O(D(log(max(s_{max}, p) + 1) + qlog(Dp + 1)) + log(\alpha ps_{max}^3 D)(2q(4q + 3)log(max(Dp, C) + 2))^{a^{2q}})$ time. We also gave an algorithm (the *EELSA* algorithm) for solving the problem. We compared the EELSA algorithm with the IYA algorithm, and found that the EELSA algorithm is giving up to 12.12% improvement over the IYA algorithm.

For future work there are many interesting problems to solve. The first problem to consider is the energy efficient task scheduling problem. We are given a set of independent tasks that share a common deadline. We have to schedule them on the proposed multi-core processor model so that the energy consumed can be minimized, and at the same time each task is completed within the given deadline. It seems to be a more difficult problem than the *EELSP* problem that we discussed in this work. It will be interesting to find whether the problem is *NP-Complete*.

The second problem to consider is the generalization of the previous problem that considers the energy efficient task scheduling problem in which the tasks have precedence constraints, and also arrival times and deadlines.

The third problem to consider is to solve the *EELSP* problem on an extension of our proposed multi-core processor model that includes the case in which a core can go into a sleep state, but a constant amount of energy is required to bring back the core into a non-sleep state [8]. It will be interesting to find whether this extension makes the *EELSP* problem more difficult to solve.

References

[1] J. Fruehe, Planning Considerations for Multicore Processor Technology. Dell Power Solutions, May 2005, pp. 67–72.
[2] B. Schauer, Multicore Processors – A Necessity. ProQuest Discovery Guides, September 2008, pp. 1–14.
[3] P. Pillai, K.G. Shin, Real-time dynamic voltage scaling for low-power embedded operating systems, in: ACM Symposium on Operating Systems Principles, 2001, pp. 89–102.
[4] T. Ishihara, H. Yasuura, Voltage scheduling problems for dynamically variable voltage processors, in: Proceedings of the 1998 international symposium on low power electronics and design, 1998, pp. 197–202.
[5] H. Aydin, R. Melhem, D. Mossé, P. Mejía-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems, in: Proceedings of the 22nd IEEE Real-Time Systems Symposium, 2001, pp. 95–105.
[6] J.-J. Chen, T.-W. Kuo, C.-L. Yang, Profit-driven uniprocessor scheduling with energy and timing constraints, in: ACM Symposium on Applied Computing, ACM Press, 2004, pp. 834–840.
[7] J.-J. Chen, T.-W. Kuo, H.-I. Lu, Power-saving scheduling for weakly dynamic voltage scaling devices, in: Algorithms and Data Structures, Lecture Notes in Computer Science, vol. 3608/2005, Springer, 2005, pp. 338–349.
[8] S. Irani, S. Shukla, R. Gupta, Algorithms for power savings, in: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2003, pp. 37–46.
[9] P. Mejía-Alvarez, E. Levner, D. Mossé, Adaptive scheduling server for power-aware real-time tasks, ACM Trans. Embedded Comput. Syst. 3 (2) (2004) 284–306.
[10] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, IEEE, 1995, pp. 374–382.
[11] H.-S. Yun, J. Kim, On energy-optimal voltage scheduling for fixed-priority hard real-time systems, ACM Trans. Embedded Comput. Syst. 2 (3) (2003) 393–430.
[12] C.-Y. Yang, J.-J. Chen, L. Thiele, T.-W. Kuo, Energy-efficient real-time task scheduling with temperature-dependent leakage, in: The ACM/IEEE Conference of Design, Automation, and Test in Europe (DATE), Dresden, Germany, March 8–12, 2010, pp. 9–14.
[13] W. Liao, L. He, K.M. Lepak, Temperature and supply voltage aware performance and power modeling at microarchitecture level, IEEE Trans. CAD Integr. Circuits Syst. 24 (7) (2005) 1042–1053.
[14] Y. Liu, R.P. Dick, L. Shang, H. Yang, Accurate temperature-dependent integrated circuit leakage power estimation is easy, in: DATE, 2007, pp. 1526–1531.
[15] J.H. Anderson, 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, 2004, pp. 428–435.
[16] J.-J. Chen, H.-R. Hsu, K.-H. Chuang, C.-L. Yang, A.-C. Pang, T.-W. Kuo, Multiprocessor energy-efficient scheduling with task migration considerations, in: Proceedings of the 16th Euromicro Conference on Real-Time Systems, 2004, pp. 101–108.
[17] F. Gruian, System-level design methods for low-energy architectures containing variable voltage processors, in: Power-Aware Computing Systems, Lecture Notes in Computer Science, vol. 2008/2001, Springer, 2001, pp. 1–12.
[18] F. Gruian, K. Kuchcinski, Lenes: task scheduling for low energy systems using variable supply voltage processors, in: Proc. Asia South Pacific Design Automation Conference, 2001, pp. 449–455.
[19] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, R. Melhem. Energy aware scheduling for distributed real-time systems, in: International Parallel and Distributed Processing Symposium, 2003, pp. 21.2.
[20] Y. Zhang, X. Hu, D.Z. Chen, Task scheduling and voltage selection for energy minimization, in: Annual ACM/IEEE Design Automation Conference, 2002, pp. 183–188.
[21] D. Zhu, R. Melhem, B. Childers, Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor realtime systems, in: Proceedings of IEEE 22nd Real-Time System Symposium, 2001, pp. 84–94.
[22] C.-Y. Yang, J.-J. Chen, T.-W. Kuo, An approximation algorithm for energy-efficient scheduling on a chip multiprocessor, in: The Eighth ACM/IEEE Conference of Design, Automation, and Test in Europe (DATE), Munich, Germany, March 7–11, 2005, pp. 468–473.
[23] W.-Y. Shieh, C.-C. Pong, Energy and transition-aware runtime task scheduling for multicore processors, J. Parallel Distrib. Comput. 73 (2013) 1225–1238.

[24] S. Zhuravlev, J.C. Saez, S. Blagodurov, A. Fedorova, M. Prieto, Survey of energy-cognizant scheduling techniques, IEEE Trans. Parallel Distrib. Syst. 24 (2013) 1447–1464.
[25] X Chen, Z. Xu, H. Kim, P.V. Gratz, J. Hu, M. Kishinevsky, U. Ogras, R. Ayoub, Dynamic voltage and frequency scaling for shared resources in multicore processor designs, in: Proceedings of DAC'13, 2013, p. 114.
[26] N. Min-Allah, H. Hussain, S.U. Khan, A.Y. Zomaya, Power efficient rate monotonic scheduling for multi-core systems, J. Parallel Distrib. Comput. 72 (2012) 48–57.
[27] M. Etinski, J. Corbalan, J. Labarta, M. Valero, Understanding the future of energy-performance trade-off via DVFS in HPC environments, J. Parallel Distrib. Comput. 72 (2012) 579–590.
[28] X. Zhu, C. He, K. Li, X. Qin, Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters, J. Parallel Distrib. Comput. 72 (2012) 751–763.
[29] W.Y. Lee, Energy-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors, IEEE Trans. Parallel Distrib. Syst. 23 (2012) 530–537.
[30] G. Dhiman, G. Marchetti, T. Rosing, vGreen: a system for energy efficient computing in virtualized environments, in: Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design, 2009, pp. 243–248.
[31] R. Bergamaschi, G. Han, A. Buyuktosunoglu, H. Patel, I. Nair, G. Dittmann, G. Janssen, N. Dhanwada, Z. Hu, P. Bose, J. Darringer, Exploring power management in multi-core systems, in: Design Automation Conference, vol. 2008, 2008, pp. 708–713.
[32] Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor, White Paper, Intel Corporation, March 2004.
[33] AMD PowerNow! Technology, Informational White Paper, Advanced Micro Devices Inc, 2000.
[34] J. Casazza, First the Tick, Now the Tock: Intel(R) Microarchitecture (Nehalem), Intel Corporation, 2009.
[35] A. Chandrakasan, S. Sheng, R. Broderson, Lower-power CMOS digital design, IEEE J. Solid-State Circuit 27 (4) (1992) 473–484.
[36] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced CPU energy, in: Proceedings of Symposium on Operating Systems Design and Implementation, 1994, pp. 13–23.
[37] M. Fleischmann, Crusoe LongRun Power Management: Dynamic Power Management for Crusoe Processors, 2001.
[38] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, A. Nicolau, Profile-based dynamic voltage scheduling using program checkpoints, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition, 2002, pp. 168–175.
[39] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[40] H.W. Lenstra Jr., Integer Programming with a Fixed Number of Variables, Math. Oper. Res. 8 (4) (1983) 538–548.
[41] E. Horowitz, S. Sahni, S. Rajasekaran, Fundamentals of Computer Algorithms, W.H. Freeman, 1998.