An Approximation Scheme for Energy-Efficient Scheduling of Real-Time Tasks in Heterogeneous Multiprocessor Systems

Chuan-Yue Yang†, Jian-Jia Chen‡, Tei-Wei Kuo†, Lothar Thiele‡
†Department of Computer Science and Information Engineering, National Taiwan University
‡Computer Engineering and Networks Laboratory (TIK), ETH Zürich
Email: †{r92032, ktw}@csie.ntu.edu.tw, ‡{chen, thiele}@tik.ee.ethz.ch

***Abstract*—As application complexity increases, modern embedded systems have adopted heterogeneous processing elements to enhance the computing capability or to reduce the power consumption. The heterogeneity has introduced challenges for energy efficiency in hardware and software implementations. This paper studies how to partition real-time tasks on a platform with heterogeneous processing elements (processors) so that the energy consumption can be minimized. The power consumption models considered in this paper are very general by assuming that the energy consumption with higher workload is larger than that with lower workload, which is true for many systems. We propose an approximation scheme to derive near-optimal solutions for different hardware configurations in energy/power consumption. When the number of processors is a constant, the scheme is a fully polynomial-time approximation scheme (FPTAS) to derive a solution with energy consumption very close to the optimal energy consumption in polynomial-time/space complexity. Experimental results reveal that the proposed scheme is very effective in energy efficiency with comparison to the state-of-the-art algorithm.**

Keywords: Multiprocessor scheduling, Heterogeneous multiprocessor, Energy-efficient scheduling.

I. Introduction

To prolong the battery lifetime in battery-driven embedded systems and to reduce the electricity cost of server systems, low-power and energy-efficient designs have been one of the active areas in the past decade in both academics and industry. Adopting micro-architectural techniques, system designers can apply dynamic power management (DPM) and dynamic voltage scaling (DVS) to adjust the system mode and the supply voltage dynamically for energy saving, respectively. In addition, to conquer the dramatic increasing power density of electronic circuits, multiprocessor platforms have been adopted for the improvement of performance without incurring too much energy overhead. As real-time constraints are required to maintain the stability of systems, one of the key issues in embedded systems is to minimize the energy consumption under the timing requirements.

Energy-efficient task scheduling/partition in homogeneous multiprocessor systems has been studied extensively in the literature, e.g., [1], [3]–[5], [7], [15], [16], [18]. However, only few results have been developed for energy-efficient task partition in heterogeneous multiprocessor systems. Specifically, Yu and Prasanna [17] proposed a heuristic algorithm based on *Integer Linear Programming* (ILP) for processors with discrete speeds. Huang, Tsai, and Chu [9] developed a greedy algorithm based on affinity to assign frame-based real-time tasks with re-assignment in pseudo polynomial-time to minimize the energy consumption when any processing speed can be assigned for a processor. Luo and Jha [14] developed heuristics based on the list-scheduling strategy for tasks with precedence constraints in heterogeneous distributed systems. Unfortunately, the above research results for energy consumption minimization in heterogeneous multiprocessor systems do not provide guarantees on quality or feasibility of the derived solutions. To our best knowledge, the approaches by Hung, Chen, and Kuo [10] for platforms with one DVS processor and one non-DVS processor are the only existing algorithms with worst-case guarantees in energy consumption minimization or energy saving maximization.

For the sake of effective use of multiprocessor environments, this paper explores energy-efficient task partitioning of real-time tasks in a system with heterogeneous processors (or processing elements). The objective is to derive schedules that minimize the energy consumption without violating the timing constraints. We consider both dynamic power consumption and static power consumption of the processors, while most existing researches for heterogeneous multiprocessor systems, such as [6], [8]–[10], [17], only consider dynamic power consumption. We assume that the energy consumption with higher workload is larger than that with lower workload. For example, systems with periodic real-time tasks without DPM or systems with frame-based real-time tasks with DPM for discrete/continuous available speeds both satisfy the above assumption, in which frame-based real-time tasks have the same period/deadline. This general energy consumption model covers the models used in [6], [8]–[10], [17].

By applying the dynamic programming approach and trimming some states by rounding, we develop an approximation scheme to derive near-optimal solutions for different hardware configurations in energy/power consumption. The proposed scheme is proved as a fully polynomial-time approximation scheme (FPTAS) when the number of processors is a constant. System designers can specify a parameter for trading the quality of the derived solution, in terms of energy consumption, to the running time of the algorithm. It takes more time/space complexity for deriving a more energy-efficient solution. Performance evaluations for heterogeneous DVS multiprocessor systems with comparison to the state-of-the-art approach [9] show that our derived solutions could improve $10\% \sim 15\%$ when static/leakage power consumption is negligible and improve $30\% \sim 60\%$ when static power consumption is non-negligible.

The rest of this paper is organized as follows: Section II defines the system models and the studied problem for systems with heterogeneous processors. Section III presents energy-efficient algorithms for task partition. The simulation results for performance evaluation are presented in Section IV. Section V concludes this paper.

II. Systems Models

This section presents the processor models, task models, energy models, and problem definitions.

This research was done while Chuan-Yue Yang was visiting TIK, ETH Zürich, and is supported in part by grants from ROC National Science Council 97-2221-E-002-206-MY3, 95-2221-E-002-094-MY3, NSC-096-2917-I-564-121, and the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 216008.

A. Processor Models

We explore energy-efficient scheduling for systems equipped with M heterogeneous (unrelated) processing elements, abbreviated as PEs, or processors, where each PE has its own power characteristics and processing capability. We assume that the number of PEs, denoted by M, is a *fixed constant*. These M given PEs are named as $m_1, m_2, \ldots, m_M$. The power consumption function $P_j(s)$ of PE m_j at the adopted processing speed s can be divided into two parts $P_j^{dep}(s)$ and P_j^{ind}, where $P_j^{dep}(s)$ and P_j^{ind} are dependent and independent on the adopted processing speed, respectively [19].

The dynamic power and the short-circuit power are the major contributions towards the speed-dependent power consumption function $P_j^{dep}(s)$. The speed-independent power consumption P_j^{ind} mainly comes from the leakage power consumption. If the leakage power consumption is related to the adopted processing speed, it could be divided into two parts and contribute to $P_j^{dep}(s)$ and P_j^{ind} accordingly.

For PEs under micro-meter manufacturing, the speed-dependent power consumption dominates the power consumption of a PE. As the speed-independent power consumption becomes comparable to the speed-dependent power consumption for PEs with nano-metro manufacturing, some dynamic power management techniques might be adopted to reduce the energy consumption by turning on/off PEs dynamically. The energy overhead, denoted by $E_j^{on} \geq 0$, to turn on/off PE m_j depends on its cache size and some hardware characteristics.

As we focus on systems with heterogeneous PEs, the processing capabilities of PEs might be different from each other. This means some applications/tasks might require much less computation in some specific PEs than others. We assume each PE has its own voltage island, and can adjust its processing speed independently. The maximum (minimum, respectively) available speeds of PE m_j is s_j^{max} (s_j^{min}, respectively). For the *ideal* DVS PE m_j, any speed in the range of $[s_j^{min}, s_j^{max}]$ can be used for execution. For the *non-ideal* DVS model, the PE can only use discrete speeds $s_j^{min} = s_{j,1} < s_{j,2} < \cdots < s_{j,k_j} = s_j^{max}$ for execution, where k_j is the number of available speeds on PE m_j. Without idling, the number of cycles executed over the processor m_j in an interval $(t_1, t_2]$ is $\int_{t_1}^{t_2} s_j(t)dt$, and the energy consumption is $\int_{t_1}^{t_2} P_j(s_j(t))dt$, where $s_j(t)$ is the adopted processing speed at time t on PE m_j.

B. Task Models

The tasks executed on the target platform is a task set $\mathbf{T}$ consisting of N tasks, $\tau_1, \tau_2, \ldots, \tau_N$. Executing task τ_i on PE m_j requires $c_{i,j}$ execution time at speed s_j^{max} in the worst-case. We consider periodic real-time tasks without precedence constraints, in which each task τ_i has its period p_i and relative deadline p_i. The workload of task τ_i while it is assigned to PE m_j can then be characterized by its utilization at the maximum speed $u_{i,j} = \frac{c_{i,j}}{p_i}$.

Specifically, if all the tasks have the same period D, i.e., $p_i = D$ for all task $\tau_i \in \mathbf{T}$ with the same arrival time, the task set is so-called frame-based task set. Moreover, as shown in [13], for frame-based real-time tasks with precedence constraints, one could apply pipeline scheduling to transform the problem as tasks without precedence constraints. Section II-C will explain how to evaluate the energy consumption of periodic real-time tasks and frame-based real-time tasks.

To schedule these tasks, global scheduling or partition scheduling could be adopted. This paper explores how to perform partition scheduling, in which each task is assigned to one PE for execution. Moreover, since energy-efficient scheduling for periodic real-time tasks on a processor is optimal by applying the earliest-deadline-first (EDF) strategy [2], for the rest of this paper, we simply use EDF for task scheduling after partitioning tasks in $\mathbf{T}$.

C. Energy Consumption

Suppose that $E_j(U_j)$ is the energy consumption when U_j is the workload assigned to PE m_j without violating the timing constraints. The energy consumption function $E_j(U_j)$ can be defined by any workload U_j in the range of $(0, 1]$. Without loss of generality, if no task is assigned to PE m_j, i.e., $U_j = 0$, we can simply set the energy consumption of m_j as 0 (or a constant). Moreover, if $U_j > 1$, $E_j(U_j)$ is set to ∞.

There is no assumption on the energy consumption model for systems which our proposed algorithm is applicable to. However, our proposed algorithm works with performance guarantees if all functions $E_j()$'s confirm the following equation:

$$E_j((1+\sigma)U_j) \leq E_j((1+\delta)U_j) \leq (1+\epsilon)E_j(U_j), \qquad (1)$$

for any $U_j > 0$, $(1+\delta)U_j \leq 1$, $0 \leq \sigma \leq \delta$, where δ is a polynomial function of constant ϵ in a specified range. Equation (1) means that

1) the energy consumption on PE m_j is a non-decreasing function of U_j, and
2) the energy consumption on PE m_j of workload $(1+\delta)U_j$ is no more than that of workload U_j times $(1+\epsilon)$.

Once the energy consumption of a PE satisfies Equation (1), the energy consumption is non-decreasing and the variance is limited when the workload is increased. With such a property of the energy consumption, our algorithm can provide worst-case guarantees in energy consumption minimization. For the rest of this subsection, we are going to present those models in which Equation (1) holds.

Ideal DVS: If there is no DPM for reducing the speed-independent power consumption, we can simply assume that P_j^{ind} is a constant. For energy-efficient scheduling of periodic real-time tasks in such a case, as shown by Aydin et al. [2], an optimal solution is to execute at a constant speed such that the utilization is either 100% or at the minimum speed with utilization less than 100%. Namely, the energy consumption $E_j(U_j)$ is

$$E_j(U_j) = \begin{cases} 0 & \text{if } U_j = 0, \\ L(P_j^{dep}(s_j^{min}) + P_j^{ind}) & \text{if } 0 < U_j \leq \frac{s_j^{min}}{s_j^{max}}, \\ L(P_j^{dep}(U_j s_j^{max}) + P_j^{ind}) & \text{if } \frac{s_j^{min}}{s_j^{max}} < U_j \leq 1, \end{cases} \qquad (2)$$

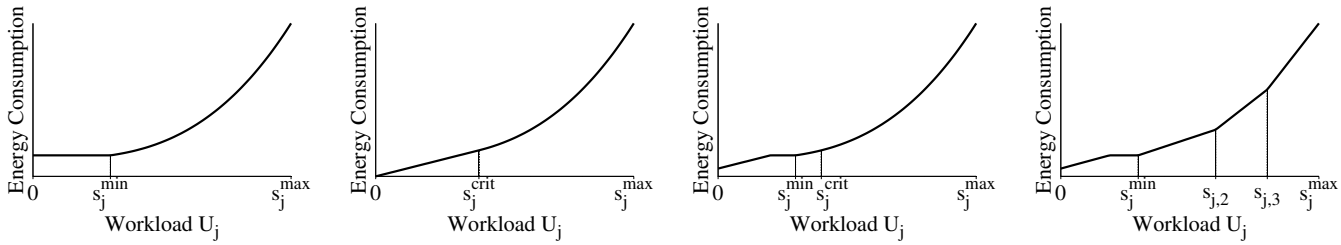
where L is the interval for evaluating the energy consumption.[1] Figure 1(a) illustrates an example, when $P_j(s) = s^3 + P_j^{ind}$ for some constant P_j^{ind}.

If we can apply DPM without any energy/timing overhead to turn on/off the PE, we might have to execute at the critical speed to reduce the energy consumption [4], [12]. The *critical speed* s_j^{crit} on PE m_j is defined as the available speed with the minimum energy consumption for execution on m_j. That is, $P_j(s_j^{crit})/s_j^{crit} \leq P_j(s)/s$ for any $s_j^{min} \leq s \leq s_j^{max}$. For such systems, as shown in [4],

$$E_j(U_j) = \begin{cases} 0 & \text{if } U_j = 0, \\ LP_j(s_j^{crit})\frac{U_j s_j^{max}}{s_j^{crit}} & \text{if } 0 < U_j \leq \frac{s_j^{crit}}{s_j^{max}}, \\ LP_j(U_j s_j^{max}) & \text{if } \frac{s_j^{crit}}{s_j^{max}} < U_j \leq 1. \end{cases} \qquad (3)$$

As a result, it is clear that the conditions in Equation (1) are satisfied for such systems. Figure 1(b) presents an example for Equation (3).

[1] If the hyper-period of these tasks exists, L is the hyper-period.



(a) Ideal DVS, $s_j^{min} > 0$ (b) Ideal DVS, $s_j^{crit} > 0$ (c) Ideal DVS, $E_j^{on} > 0$ (d) Non-ideal DVS, $E_j^{on} > 0$

Fig. 1. Examples of energy consumption functions $E_j(U_j)$ when s_j^{max} is normalized to 1.

If it takes non-negligible energy overhead E_j^{on} of turning on/off PE m_j, unfortunately, periodic real-time tasks have no such good properties of Equation (1) since we cannot define $E_j(U_j)$ due to the ignoring of period/deadline in the definition of $E_j()$.[2] However, the energy consumption function for frame-based real-time tasks with common deadline D can still satisfy Equation (1). Specifically, if we apply workload-conserving scheduling (the processor is never idle when there is a job in the ready queue), the energy consumption function $E_j(U_j)$ in Equation (3) is revised to $\min\left\{DP_j(s_j^{crit})\frac{U_j s_j^{max}}{s_j^{crit}} + E_j^{on}, DP_j\left(\max\left\{s_j^{min}, U_j s_j^{max}\right\}\right)\right\}$ when $0 < U_j \leq \frac{s_j^{crit}}{s_j^{max}}$, where L in Equation (3) is revised to D. Figure 1(c) shows an example of the energy consumption function $E_j(U_j)$ with non-negligible E_j^{on}.

Non-Ideal DVS: For a non-ideal DVS PE m_j, if $U_j s_j^{max}$ is not an available speed on m_j, we can simply use two nearest adjacent available discrete speeds to satisfy the timing constraints [11]. Thus, when the speed switching overhead is negligible, the energy consumption function $E_j(U_j)$ of a non-ideal DVS PE m_j can be defined by (a.) revising $E_j(U_j)$ as $L(P_j(s_{j,k})\frac{s_{j,k+1}-U_j s_j^{max}}{s_{j,k+1}-s_{j,k}} + P_j(s_{j,k+1})\frac{U_j s_j^{max}-s_{j,k}}{s_{j,k+1}-s_{j,k}})$ for all $\frac{s_{j,k}}{s_j^{max}} < U_j \leq \frac{s_{j,k+1}}{s_j^{max}}, 1 \leq k < k_j$ and (b.) replacing critical speed s_j^{crit} with the available discrete speed on m_j that has the minimum energy consumption for executing in Equations (2) and (3). Figure 1(d) shows an example of such an energy consumption function, where $k_j = 4$ and $E_j^{on} > 0$. As a result, the conditions in Equation (1) are still satisfied.

When the speed switching overhead is not negligible, the energy consumption function of a PE for workload of a frame-based real-time task set can also satisfy the conditions in Equation (1) with some constants ϵ and δ depending on the hardware and period D. However, when the speed switching overhead is not negligible, the energy consumption function of a PE for workload of periodic tasks can not satisfy the conditions in Equation (1). This is similar to the effect of energy overhead of turning on/off an ideal DVS PE for scheduling periodic tasks.

D. Problem Definition

Given an input task set $\mathbf{T}$ of N tasks on M heterogeneous PEs, a task partition, abbreviated as partition, Θ is *feasible* if $\mathbf{T}_\ell^\Theta \bigcap \mathbf{T}_j^\Theta = \emptyset$ $\forall \ell \neq j$, $\mathbf{T}_1^\Theta \bigcup \mathbf{T}_2^\Theta \bigcup \ldots \bigcup \mathbf{T}_M^\Theta = \mathbf{T}$, and $\sum_{\tau_i \in \mathbf{T}_j^\Theta} u_{i,j} \leq 1$ $\forall 1 \leq j \leq M$, where $\mathbf{T}_j^\Theta$ is the set of tasks assigned to PE m_j in the task partition Θ. In addition, given energy consumption function $E_j(U_j)$ for each PE m_j satisfying Equation (1), the objective of the studied problem is to find an *optimal* task partition Θ with the minimum energy consumption among all feasible task partitions.

As the studied problem is $\mathcal{NP}$-hard, the objective of this paper is to derive approximate solutions. An algorithm is a ρ-approximation algorithm for the problem if the derived solution has energy consumption no more than ρ times of the energy consumption of an optimal solution. Moreover, a fully polynomial-time approximation scheme (FPTAS) is a $(1+\epsilon)$-approximation algorithm with polynomial-time complexity by treating $\frac{1}{\epsilon}$ as an input parameter for any ϵ in specified ranges. Unless $\mathcal{NP} = \mathcal{P}$, fully polynomial-time approximation schemes are the best in terms of polynomial-time approximation algorithms with worst-case guarantees.

Note that for energy consumption functions that do not satisfy Equation (1), the proposed scheme can still be applied. Unfortunately, there is no worst-case guarantee of the quality in terms of energy consumption of the derived solutions.

III. Algorithms for Task Partition

This section first presents a dynamic programming for deriving the optimal task partition for the studied problem in exponential time complexity. The idea behind the dynamic programming is then used to design our fully polynomial-time approximation scheme.

A. A Dynamic Programming

The basic idea of the dynamic programming is to keep tracing a set of states $\mathcal{S}$, where each state stands for a partition Θ for a subset of task set $\mathbf{T}$. The dynamic programming considers tasks in $\mathbf{T}$ one by one. At the end, the final set of states will consist of several states, including optimal task partitions. What we have to do is to pick an optimal task partition in the final set of states as the solution.

More specifically, a state that stands for a partition Θ is presented by a tuple $\theta = (U_1^\theta, U_2^\theta, \ldots, U_M^\theta)$, where $U_j^\theta = \sum_{\tau_i \in \mathbf{T}_j^\Theta} u_{i,j}$. Initially, as none task has been considered, the initial set of states $\mathcal{S}_0$ only consists of one state $\theta = (0, 0, \ldots, 0)$. Then the dynamic programming takes each task in $\mathbf{T}$ into consideration in an arbitrary order. When the ith task τ_i is considered, a set $\mathcal{S}_i$ of new states will be generated according to the existing states in $\mathcal{S}_{i-1}$. For each state θ in $\mathcal{S}_{i-1}$, the dynamic programming constructs M new states $(U_1^\theta + u_{i,1}, U_2^\theta, \ldots, U_M^\theta)$, $(U_1^\theta, U_2^\theta + u_{i,2}, \ldots, U_M^\theta)$, …, $(U_1^\theta, U_2^\theta, \ldots, U_M^\theta + u_{i,M})$ and adds them into set $\mathcal{S}_i$. After all the N tasks are considered, all feasible task partitions are in the set $\mathcal{S}_N$. Thus, an optimal solution can be obtained by evaluating task partitions in $\mathcal{S}_N$ and picking the one which has the minimum total energy consumption.

It is clear that the dynamic programming can be improved by discarding some dominated states. A state θ_ℓ is dominated by another state θ_q if $U_j^{\theta_q} \leq U_j^{\theta_\ell}$ for $\forall j = 1, 2, \ldots, M$. Dominated states can be discarded without affecting the optimality of the derived

[2]Since periodic tasks are considered, the procedure to turn off can be assumed instantaneously with negligible energy overhead by treating the overhead as a part of the overhead to turn on the PE.

solutions, since energy consumption functions for PEs are non-decreasing functions of workloads. Although the derived solution of the above dynamic programming is optimal in energy consumption minimization, it requires exponential time and space complexity, i.e., $O(M^N)$. Therefore, more efficient implementation is required to derive near-optimal solutions in polynomial time.

B. An FPTAS

The reason why the dynamic programming in Section III-A is with exponential time and space complexity is that there are too many states in the sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N$.Thus, in order to improve the efficiency of the dynamic programming to obtain a fully polynomial-time approximation scheme, we have to prune some states in the sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N$ so that the number of states does not increase exponentially in the input size. Moreover, state pruning must be done carefully to limit the sacrifice in the optimality of the final result. We use a user-specified constant ϵ to achieve to above contradicted objectives such that the time complexity is polynomial in the input size and $\frac{1}{\epsilon}$ and the approximation factor is $1+\epsilon$.

Given a user-specified constant ϵ with an input instance, we first find a constant δ_j for each PE m_j such that $E_j((1+\delta_j)U_j) \leq (1+\epsilon)E_j(U_j)$ for any $(1+\delta_j)U_j$ in $(0, 1]$. As shown in Section II-C, since $E_j()$ is a function satisfying the conditions in Equation (1), the constant δ_j must exist and is polynomial in ϵ. Such a constant δ_j can be obtained from the observation on the curve of $E_j()$ and a binary search, or from the algebraic calculation for some cases. For example, when $P_j(s) = s^\alpha + P_j^{ind}$ for some $1 \leq \alpha \leq 3$ and constant P_j^{ind}, δ_j can be set as $\frac{\epsilon}{7}$ when $0 < \epsilon \leq 7$. After δ_j is determined, we define another constant $\gamma_j = \frac{\ln(1+\delta_j)}{N}$ that will be used when we round down the workload of the states on PE m_j.

The basic idea of the pruning process is to round down the workload U_j on each PE m_j except that on PE m_1 for all states when constructing sets $\mathcal{S}_i$.[3] If two states θ_ℓ and θ_q agree with all rounded workloads for $m_2, m_3, \ldots, m_M$, either θ_ℓ dominates θ_q or θ_q dominates θ_ℓ, and hence, we can discard one of them. Assuming that the number of distinct workload U_j after rounding down is n_j for all $2 \leq j \leq M$, the number of different (rounded) states in $\mathcal{S}_i$ is at most $K = \prod_{2 \leq j \leq M} n_j$. Therefore, the principle for designing a polynomial time algorithm is to make sure that n_j is polynomial in the input size and $\frac{1}{\epsilon}$ for all $2 \leq j \leq M$. Of course, the more states kept after rounding down usually lead to a more precise solution.

Algorithm 1, denoted by Algorithm MTRIM, sketches our proposed approximation scheme. For each iteration to consider task τ_i, we first create a temporary set $\mathcal{S}'$ of states, and then round down the workloads of the states in $\mathcal{S}'$ into a set $\mathcal{S}_i$ of rounded states by using Procedure PRUNE in Algorithm 1. After considering all the tasks in $\mathbf{T}$, we choose the state θ^* in $\mathcal{S}_N$ with the minimum (rounded) energy consumption as the solution. As θ^* is a rounded state, we can then determine a corresponding task partition Θ^* by backtracking. The detail of the rounding process for workloads of PE m_j is presented in Procedure PRUNE in Algorithm 1.

In Procedure PRUNE, we first round down the workloads on $m_M, m_{M-1}, \ldots, m_2$ from Step 1 to Step 9, and then choose the (rounded) state with the minimum workload on PE m_1 from Step 11 to Step 14 to represent all the states agreeing with all workloads on PE m_j for all $2 \leq j \leq M$. For rounding workloads on m_j with $j \geq 2$, the details are presented from Step 2 to Step 8 of Procedure PRUNE. In Step 2, we first sort states in $\mathcal{S}$ according to U_j^θ in a non-decreasing order, says $\theta_1, \theta_2, \ldots, \theta_{|\mathcal{S}|}$. Then in Step 3, $U_j^{\theta_1}$ is set as the first distinct workload U_j^θ. All remaining states in $\mathcal{S}$ are checked one by one according to the sorted order. If the workload $U_j^{\theta_\ell}$ of a state θ_ℓ is no greater than $(1+\gamma_j)U_j^\theta$, it is rounded to U_j^θ. Otherwise, the workload $U_j^{\theta_\ell}$ of the state θ_ℓ is chosen as the next distinct workload U_j^θ, and the checking continues for the remaining states until all states in $\mathcal{S}$ have been checked. Suppose that there are n_j distinct workloads named $U_j^{\hat{\theta}_1}, U_j^{\hat{\theta}_2}, \ldots, U_j^{\hat{\theta}_{n_j}}$ after the rounding process for workloads on PE m_j, where $U_j^{\hat{\theta}_{i+1}} > (1+\gamma_j)U_j^{\hat{\theta}_i}$ for all $i < n_j$.

[3]One could also round down U_1, but it makes the complexity higher.

Algorithm 1 : MTRIM

Input: $(\mathbf{T}, \{E_j()\}, M, \epsilon)$;
1: determine δ_j according to $E_j()$ and ϵ, and let $\gamma_j = \frac{\ln(1+\delta_j)}{N}$ for each PE m_j;
2: $\mathcal{S}_0 \leftarrow \{(0, 0, \ldots, 0)\}$;
3: **for** $i \leftarrow 1$ to N **do**
4: $\quad \mathcal{S}' \leftarrow \{(U_1^\theta + u_{i,1}, U_2^\theta, \ldots, U_M^\theta), (U_1^\theta, U_2^\theta + u_{i,2}, \ldots, U_M^\theta), \ldots, (U_1^\theta, U_2^\theta, \ldots, U_M^\theta + u_{i,M}) | \theta \in \mathcal{S}_{i-1}\}$;
5: $\quad$ remove state θ from $\mathcal{S}'$ where there exists $U_j^\theta > 1$;
6: $\quad \mathcal{S}_i \leftarrow$ PRUNE$(\mathcal{S}', M)$;
7: find the state θ^* in $\mathcal{S}_N$ with minimum $\sum_{j=1}^M E_j(U_j^{\theta^*})$;
8: return the corresponding task partition Θ^* of the state θ^*;

Procedure: PRUNE$(\mathcal{S}, j)$;
1: **if** $j \geq 2$ **then**
2: $\quad$ sort states in $\mathcal{S}$ as $\theta_1, \theta_2, \ldots, \theta_{|\mathcal{S}|}$ so that $U_j^{\theta_\ell} \leq U_j^{\theta_q}$ if $\ell < q$;
3: $\quad \theta \leftarrow \theta_1$;
4: $\quad$ **for** $\ell \leftarrow 2$ to $|\mathcal{S}|$ **do**
5: $\quad\quad$ **if** $U_j^{\theta_\ell} \leq (1+\gamma_j) \cdot U_j^\theta$ **then**
6: $\quad\quad\quad U_j^{\theta_\ell} \leftarrow U_j^\theta$;
7: $\quad\quad$ **else**
8: $\quad\quad\quad \theta \leftarrow \theta_\ell$;
9: $\quad$ return PRUNE$(\mathcal{S}, j-1)$;
10: **else**
11: $\quad$ divide $\mathcal{S}$ into subsets so that $U_y^{\theta_\ell} = U_y^{\theta_q}$ for $\forall y = 2, 3, \ldots, M$ for all θ_ℓ, θ_q in the same divided subset;
12: $\quad \mathcal{S}^\flat \leftarrow \emptyset$;
13: $\quad$ for each of the divided subset of $\mathcal{S}$, find one state θ that has the minimal U_1^θ and add it into $\mathcal{S}^\flat$;
14: $\quad$ return $\mathcal{S}^\flat$;

After presenting Algorithm MTRIM, we now show that the algorithm is with polynomial time complexity in the input size and $\frac{1}{\epsilon}$ and has an approximation factor $1+\epsilon$. For the following analysis, we assume that the derived task partition is feasible. At the end of this section, we will present how to deal with the case that the derived solution is infeasible. For brevity, let Θ^* be the task partition derived from Algorithm MTRIM and θ^* be the corresponding rounded state. The following lemma provides the approximation ratio of Algorithm MTRIM when the derived solution Θ^* is a feasible task partition of $\mathbf{T}$.

Lemma 1: The total energy consumption of the solution derived from Algorithm MTRIM is no greater than $(1+\epsilon)$ times of that of an optimal solution if the derived task partition is feasible.

Proof: It is not difficult to see that $\sum_{j=1}^M E(U_j^{\theta^*})$ is a lower bound of the energy consumption of an optimal solution. For task τ_i, when we invoke Procedure PRUNE$(\mathcal{S}', M)$, the workload of a state might be rounded down by at most a factor of $(1+\gamma_j)$ on PEs m_j for all $2 \leq j \leq M$. Since Procedure PRUNE$(\mathcal{S}', M)$ is invoked N times in Algorithm MTRIM, the actual workload $U_j = \sum_{\tau_i \in \mathbf{T}_j^{\Theta^*}} u_{i,j}$ on

PE m_j for the task partition Θ^* is at most $(1+\gamma_j)^N$ times of $U_j^{\theta^*}$. Moreover,

$$(1+\gamma_j)^N = \left(1+\frac{\ln(1+\delta_j)}{N}\right)^N \leq e^{\ln(1+\delta_j)} = 1+\delta_j.$$

Therefore, we have $E_j(U_j) \leq E_j((1+\gamma_j)^N U_j^\theta) \leq E_j((1+\delta_j)U_j^\theta) \leq (1+\epsilon)E_j(U_j^\theta)$, where the last inequality comes directly from the setting of δ_j in Algorithm MTRIM. Thus, the energy consumption of each PE is no greater than $(1+\epsilon)$ times of that in a lower bound of an optimal solution, so is the total energy consumption. ■

Then Lemma 2 shows the number of states in $\mathcal{S}_i$ is polynomial in the input size and $\frac{1}{\epsilon}$.

Lemma 2: The number of states in $\mathcal{S}_i$ is at most $K = O(N^{M-1} \cdot \prod_{2\leq j\leq M} \frac{1}{\delta_j}\log\lambda_j)$ and is polynomial in the input size and $\frac{1}{\epsilon}$, where λ_j is $\frac{\min\{1,\sum_{l=1}^N u_{l,j}\}}{\min_{\tau_l\in\mathbf{T}} u_{l,j}}$ for $\forall 1 \leq i \leq N$.

Proof: Let $U_j^{\hat{\theta}_1}, U_j^{\hat{\theta}_2}, \ldots, U_j^{\hat{\theta}_{n_j}}$ be the n_j distinct workloads on PE m_j in an increasing order after rounding the workloads on PE m_j in Procedure PRUNE. If $U_j^{\hat{\theta}_1} > 0$, we know that $U_j^{\hat{\theta}_{n_j}} > (1+\gamma_j)^{n_j-1} U_j^{\hat{\theta}_1}$. Thus,

$$(1+\gamma_j)^{n_j-1} < \frac{U_j^{\hat{\theta}_{n_j}}}{U_j^{\hat{\theta}_1}} \leq \frac{\min\{1,\sum_{l=1}^N u_{l,j}\}}{\min_{\tau_l\in\mathbf{T}}\{u_{l,j}\}} = \lambda_j.$$

Similarly, we have $(1+\gamma_j)^{n_j-2} < \lambda_j$ when $U_j^{\hat{\theta}_1} = 0$. Therefore,

$$\begin{aligned} n_j &< 2 + \tfrac{\ln\lambda_j}{\ln(1+\gamma_j)} \leq 2 + \tfrac{\ln\lambda_j}{\frac{\gamma_j}{1+\gamma_j}} \\ &= 2 + \ln\lambda_j\left(1+\tfrac{N}{\ln(1+\delta_j)}\right) = O(N\tfrac{\log\lambda_j}{\delta_j}), \end{aligned}$$

where the second inequality comes from $\ln x \geq \frac{x-1}{x}$ when $x > 1$. The number of states in $\mathcal{S}_i$ is the same as the number of the divided subsets constructed in Step 11 of Procedure PRUNE. It is equal to $\prod_{2\leq j\leq M} n_j$ and can be bounded by $K = O(N^{M-1} \cdot \prod_{2\leq j\leq M} \frac{\log\lambda_j}{\delta_j})$, where K is polynomial in $N, \frac{1}{\epsilon}$ and the number of bit required to encode the input if M is considered as a constant. ■

By applying the above lemmas, we can show that Algorithm MTRIM is a fully polynomial-time approximation scheme if the derived solution is feasible.

Theorem 1: Algorithm MTRIM is a fully polynomial-time approximation scheme for the energy-efficient task partition problem when the derived solution is feasible and the number of PEs M is a constant.

Proof: The time complexity of Algorithm MTRIM is dominated by the loop from Step 3 to Step 6 and each iteration is dominated by the rounding process in Procedure PRUNE. By Lemma 2, when we consider task τ_i, there are KM states in set $\mathcal{S}'$, where $K = O(N^{M-1} \cdot \prod_{2\leq j\leq M} \frac{1}{\delta_j}\log\lambda_j)$. As we will sort these states in set $\mathcal{S}'$ for $O(M)$ times, the time complexity for Procedure PRUNE is $O(M \cdot KM\log(KM))$. Thus, the overall time complexity of Algorithm MTRIM is $O(N \cdot M \cdot KM\log(KM))$. According to Lemma 2, we know it is polynomial in $N, \frac{1}{\epsilon}$ and the number of bit required to encode the input while M is a constant. Moreover, as the derived solution is at most $1+\epsilon$ times of the optimal solution when it is feasible, we conclude the proof. ■

Remarks: When the solution Θ^* derived from Algorithm MTRIM is not a feasible task partition for $\mathbf{T}$, we should be a little bit careful. We can sort all states θ's in $\mathcal{S}_N$ in terms of the rounded total energy consumption ($\sum_{j=1}^M E_j(U_j^\theta)$). Then, starting from the state with the minimum rounded total energy consumption, we backtrack its corresponding task partition and check if the task partition is feasible. If it is feasible, we return the solution; otherwise, we try the next state repetitively. If there exist some feasible task partitions in $\mathcal{S}_N$, Algorithm MTRIM can return the best one as the solution. However, we can not provide any worst-case performance guarantee for the derived solution in this case. Unfortunately, as deriving a feasible task partition is $\mathcal{NP}$-complete, unless $\mathcal{P} = \mathcal{NP}$, there does not exist any polynomial-time algorithm for deriving a feasible task partition even if feasible task partition exists.

IV. PERFORMANCE EVALUATION

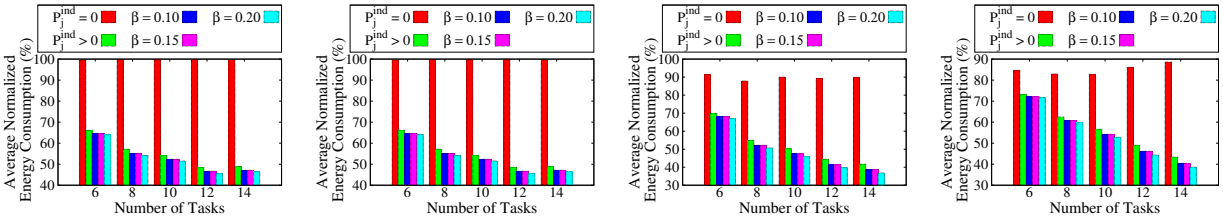
Simulation Setup: We performed a series of simulations to demonstrate the strength of our proposed approximation scheme. The simulation setup was similar to that in [9]. There were nearly 30 types of PEs, including general-purpose embedded processors, such as ARM9 and ARM11, and digital signal processors (DSP), such as TMS320C and TMS320D, in our simulations. For each configuration, we chose M PEs from the PE types to form a target hardware platform, where $M = 2, 4, 6$. Due to space limitation, we only present the simulation results for ideal DVS PEs. Moreover, to evaluate the performance for different energy consumption models, we performed simulations for periodic real-time tasks when E_j^{on} was 0 and simulations for frame-based real-time tasks with common period $D = 50ms$ when E_j^{on} was non-negligible. The number N of tasks was $N = 6, 8, 10, 12, 14$. The worst-case execution time $c_{i,j}$ of task τ_i on PE m_j was set randomly from $1000us$ to $3000us$ at speed s_j^{max}.

In addition to the models in [9], we also simulated energy consumption models with non-negligible speed-independent power consumption P_j^{ind} and energy overhead of turning on/off PE m_j, i.e., E_j^{on}. In order to make the speed-independent power consumption non-negligible in the task partition problem, P_j^{ind} should be defined according to the workload of task sets. We set $\frac{\sum_{\tau_i\in\mathbf{T}} u_{i,j}}{M} s_j^{max}$ as the critical speed s_j^{crit} of PE m_j to define P_j^{ind}, i.e., $P_j^{ind} = 2\kappa_j(s_j^{crit})^3$ when $P_j^{dep}(s) = \kappa_j s^3$ for some constant κ_j. As the energy overhead of turning on/off PE m_j depended on its cache size and some hardware characteristics, E_j^{on} was assumed in a range from $0.05 \cdot DP_j(s_j^{crit})$ mJ to $\beta \cdot DP_j(s_j^{crit})$ mJ for a constant $0.05 < \beta < 1$.

Our proposed approximation scheme was evaluated by setting ϵ as different values, compared with the greedy-based algorithm in [9]. The energy consumption of the task partition derived from Algorithm MTRIM divided by that of the task partition derived from the greedy-based algorithm in [9] was defined as the *Normalized Energy Consumption*, which is adopted as the performance metrics of our simulations. We simulated each configuration for 128 input instances independently and reported the average value.

Simulation Results: Figure 2 shows the average normalized energy consumption of Algorithm MTRIM in our simulations, where $P_j^{ind} = 0$ stands for the results of energy consumption models of $E_j(U_j) = \kappa_j(U_j s_j^{max})^3$ the same as the model in [9], $P_j^{ind} > 0$ stands for the results of energy consumption models with non-negligible speed-independent power, and $\beta = 0.10, \beta = 0.15$, and $\beta = 0.20$ stand for the results of energy consumption models with different ranges of energy overheads E_j^{on} to turning on/off PE m_j.

Figures 2(a) and 2(b) are the results by setting ϵ as 0.5 and 1, respectively, when $M = 2$. According to Figures 2(a) and 2(b), the difference between the results of $\epsilon = 1$ and that of $\epsilon = 0.5$ is less than 0.2%. Thus, it is sufficient to set ϵ as 1 for Algorithm MTRIM



(a) $M = 2$, $\epsilon = 0.5$ (b) $M = 2$, $\epsilon = 1$ (c) $M = 4$, $\epsilon = 1$ (d) $M = 6$, $\epsilon = 1$

Fig. 2. The average normalized energy consumption of Algorithm MTRIM

to obtain a near-optimal solution for the studied problem. Moreover, for the energy consumption model with negligible speed-independent power and energy overhead of turning on/off PEs, the results of Algorithm MTRIM are almost the same as that of the greedy-based algorithm proposed in [9]. This is because the derived solutions were very close to optimal values when $M = 2$. As a result, the greedy-based algorithm proposed in [9] is sufficient for this model when $M = 2$. But for the other energy consumption models, Algorithm MTRIM could improve the greedy-based algorithm proposed in [9] by at least 30%.

Figures 2(c) and 2(d) show that the results of Algorithm MTRIM by setting ϵ as 1 when $M = 4$ and 6, respectively. Note that, even for models with negligible speed-independent power and energy overhead of turning on/off PEs, Algorithm MTRIM still could improve the greedy-based algorithm proposed in [9] by $10\% \sim 15\%$ when $M \geq 4$. While the improvement for the other models is similar to that when $M = 2$.

V. CONCLUSION

This paper explores energy-efficient real-time task scheduling problems over heterogeneous multiprocessors (PEs). By adopting partition scheduling, the problem could be transformed into how to partition the input task set to the PEs such that the overall energy consumption is minimized. We propose a fully polynomial-time approximation scheme to partition the tasks when the number of PEs is a constant. Our proposed approximation scheme can be applied to wide diverse system models, in which, for each PE, (a) the energy consumption is non-decreasing when the workload is increased and (b) the energy consumption will not increase more than $(1+\epsilon)$ times when the increase of the workload is within $(1+\delta)$ times, for some constants ϵ and δ. The proposed scheme can provide certain worst-case performance guarantee when the derived solution is feasible. Hence, designers can specify a parameter for trading the quality of the derived solution, in terms of energy consumption, to the running time of the algorithm. The proposed algorithm also answers the open problem address in [10] for the existence of fully polynomial-time approximation schemes to minimize the energy consumption of two heterogeneous PEs, in which one is with DVS and the other is without DVS. The simulation results show that our approximation scheme could improve the state-of-the-art approach [9] by $10\% \sim 15\%$ when speed-independent power consumption is negligible and improve by $30\% \sim 60\%$ when speed-independent power consumption is non-negligible for ideal DVS PEs.

REFERENCES

[1] T. A. Alenawy and H. Aydin. Energy-aware task allocation for rate monotonic scheduling. In *Proceedings of the 11th IEEE Real-time and Embedded Technology and Applications Symposium (RTAS'05)*, pages 213–223, 2005.

[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] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *Proceedings of 17th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 113 – 121, 2003.

[4] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo. Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems. In *IEEE Real-time and Embedded Technology and Applications Symposium*, pages 408–417, 2006.

[5] J.-J. Chen, T.-W. Kuo, C.-L. Yang, and K.-J. King. Energy-efficient real-time task scheduling with task rejection. In *DATE*, pages 1629–1634, 2007.

[6] J.-J. Chen and L. Thiele. Energy-efficient task partition for periodic real-time tasks on platforms with dual processing elements. In *International Conference on Parallel and Distributed Systems (ICPADS)*, page 161.

[7] F. Gruian and K. Kuchcinski. Lenes: Task scheduling for low energy systems using variable supply voltage processors. In *Proceedings of Asia South Pacific Design Automation Conference*, pages 449–455, 2001.

[8] H.-R. Hsu, J.-J. Chen, and T.-W. Kuo. Multiprocessor synthesis for periodic hard real-time tasks under a given energy constraint. In *ACM/IEEE Conference of Design, Automation, and Test in Europe (DATE)*, pages 1061–1066, 2006.

[9] T.-Y. Huang, Y.-C. Tsai, and E. T.-H. Chu. A near-optimal solution for the heterogeneous multi-processor single-level voltage setup problem. In *21th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–10, 2007.

[10] C.-M. Hung, J.-J. Chen, and T.-W. Kuo. Energy-efficient real-time task scheduling for a DVS system with a non-DVS processing element. In *the 27th IEEE Real-Time Systems Symposium (RTSS)*, pages 303–312, 2006.

[11] T. Ishihara and H. Yasuura. Voltage scheduling problems for dynamically variable voltage processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 197–202, 1998.

[12] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the Design Automation Conference*, pages 275–280, 2004.

[13] H. Liu, Z. Shao, M. Wang, and P. Chen. Overhead-aware system-level joint energy and performance optimization for streaming applications on multiprocessor systems-on-chip. In *EuroMicro Conference on Real-Time Systems (ECRTS)*, pages 92–101, 2008.

[14] J. Luo and N. Jha. Static and dynamic variable voltage scheduling algorithms for realtime heterogeneous distributed embedded systems. In *the 15th International Conference on VLSI Design (VLSID'02)*, pages 719–726, 2002.

[15] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem. Energy aware scheduling for distributed real-time systems. In *International Parallel and Distributed Processing Symposium*, page 21, 2003.

[16] C.-Y. Yang, J.-J. Chen, and T.-W. Kuo. An approximation algorithm for energy-efficient scheduling on a chip multiprocessor. In *Proceedings of the 8th Conference of Design, Automation, and Test in Europe (DATE)*, pages 468–473, 2005.

[17] Y. Yu and V. K. Prasanna. Power-aware resource allocation for independent tasks in heterogeneous real-time systems. In *Proceedings of the Ninth International Conference on Parallel and Distributed Systems(ICPADS'02)*. IEEE, 2002.

[18] 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.

[19] D. Zhu. Reliability-aware dynamic energy management in dependable embedded real-time systems. In *IEEE Real-time and Embedded Technology and Applications Symposium*, pages 397–407, 2006.