Energy-Efficient Scheduling of Real-Time Tasks on Cluster-Based Multicores

Fanxin Kong†, Wang Yi‡, Qingxu Deng†
†Northeastern University, China ‡Uppsala University, Sweden
kongfx@ise.neu.edu.cn, yi@it.uu.se, dengqx@mail.neu.edu.cn

Abstract—While much work has addressed the energy-efficient scheduling problem for uniprocessor or multiprocessor systems, little has been done for multicore systems. We study the multicore architecture with a fixed number of cores partitioned into clusters (or islands), on each of which all cores operate at a common frequency. We develop algorithms to determine a schedule for real-time tasks to minimize the energy consumption under the timing and operating frequency constraints. As technical contributions, we first show that the optimal frequencies resulting in the minimum energy consumption for each island is not dependent on the workload mapped but the number of cores and leakage power on the island, when not considering the timing constraint. Then for systems with timing constraints, we present a polynomial algorithm which derives the minimum energy consumption for a given task partition. Finally, we develop an efficient algorithm to determine the number of active islands, task partition and frequency assignment. Our simulation result shows that our approach significantly outperforms the related approaches in terms of energy saving.

I. INTRODUCTION

Minimizing energy consumption is one of the most challenging topics for the design of embedded and real-time systems using multicore processors. Although the energy-efficient scheduling problem has been extensively explored for uniprocessor and multiprocessor real-time systems [1], little work has addressed multicore systems. Most previous research reduces the energy-efficient scheduling problem for multicore to that for multiprocessor systems. They all assume the per-core DVFS, i.e., each processor operates at individual frequency/voltage, and has no operating frequency constraint, e.g., [2]–[14]. On the other hand, few works limit the discussion only to the full-chip DVFS (Dynamic Voltage/Frequency Scaling) designs restricting that all the cores in one chip operate at the same clock frequency/voltage [15], [16].

Per-core DVFS provides the most flexible power management. However, as the number of cores on a chip grows [17], it’s very complex and expensive to support this strategy. On the other hand, full-chip DVFS leads to simple hardware design and implementation but limited power efficiency. To balance the trade-off between the hardware complexity and power efficiency, VFI (Voltage/Frequency Island) technique is proposed. VFI supports different voltage supplies and frequencies for different clusters on a multicore, and the cores on one chip can be partitioned into clusters, on each of which all cores operate at a common frequency [18]–[22]. Moreover, different cluster partitions represent DVFS policies of different granularity.

However, there is no research effort addressing the exact energy-efficient scheduling problem on real-time cluster-based or voltage/frequency island enabled multicore systems. Blindly adopting the existing approaches without considering the restriction and flexibility of cluster-based multicore will result in a waste of energy, thus we have to find a more general approach which can deal with not only per-core or full-chip DVFS but also DVFS policies with any granularity.

We assume a multicore architecture with a fixed number of cores partitioned into clusters on each of which all cores operate at a common frequency, and a real-time application consisting of a set of independent tasks. We shall study the optimization problem to map the real-time tasks onto the clusters, which minimizes the energy consumption under given timing constraints on the tasks and operating frequency constraints on the clusters. This is an optimization problem which has three degrees of freedom including the number of active clusters, task partition and frequency assignment. Our contributions include: (i) To the best of our knowledge, it is the first research effort that discusses the energy-efficient scheduling problem for real-time tasks on cluster-based multicore systems with non-negligible leakage power consumption from three degrees of freedom including the number of active islands, task partition and frequency assignment, which can adopt to DVFS policies with any granularity. (ii) When there are no timing constraints, we show that the operating frequencies to minimize the energy consumption of each island is only dependent on the number of cores and leakage power on the island, i.e., the cluster partition, through the concept of critical speed sequence. Moreover, each island shares a common critical speed sequence under a symmetric partitioned cluster-based multicore system. (iii) For systems with timing constraints, several efficient algorithms are proposed. An optimal polynomial-time complexity algorithm is first proposed to minimize energy consumption for a real-time cluster-based multicore system given a task partition. Then, the overall algorithm is presented to determine the number of active islands, task partition and frequency assignment. Our simulation result shows that our approaches significantly outperform the existing approaches in terms of energy saving.

II. PROBLEM SETTING

We study a multicore with the symmetric cluster partition, i.e., the number of cores in each island is the same, but the proposed algorithms can be easily adapted to the asymmetric partitions with different numbers of cores in islands.

A. VFI and Power Model

The multicore system contains $N_b \times N_c$ identical cores. $N_b$ denotes the number of islands in the multicore while $N_c$ denotes the number of cores in each island. Figure 1 shows one symmetric cluster partitions of $4 \times 4$ of a 16 cores multicore.

All cores in one island share a common voltage/frequency while those cores between islands may operate at different frequencies. Each island regulates the frequency/voltage
separately. On one extreme, each core forms an island, i.e., each core is independent of another and can adjust frequency/voltage individually, which provides the per-core DVFS policy. On the other extreme, the whole chip is an island, which provides the full-chip DVFS policy. In-between the two extremes, DVFS policies with other granularity are provided, such as 2-VFI and 4-VFI cluster partition for a multicore with 16 cores, capable of regulating the frequency/voltage separately every eight cores and four cores respectively.

Hence, to reduce leakage power, not all islands need to be shut down individually with per-core DVFS, while each island can be set off only when all cores on the island become idle for one cluster-based multicore. Even if only one core is non-idle, the whole island has to be on and consumes leakage power. Hence, to reduce leakage power, not all islands need to be active, i.e., some island may have no workload mapped on, for a cluster-based multicore system, which is also different from the full-chip DVFS where the whole chip should be always active since there is only one island.

We consider a real-time application consisting of a set of independent tasks $T = \{\tau_1, ..., \tau_N\}$ with a deadline of $D$. We focus on energy-efficient scheduling for the real-time application on a cluster-based multicore system with non-negligible leakage power consumption, where each task can be mapped on only one core. Different with previous works, the problem discussed in this paper aims at deriving a schedule for real-time tasks to minimize the energy consumption with both timing and operating frequency constraints, which has three degrees of freedom including the number of active islands, task partition and frequency assignment. Since this problem is NP-hard, the objective of this paper is to derive polynomial-time complexity heuristic solutions.

### III. Critical Speed Sequence

In this section, we consider a multicore system with no timing constraint. Due to the non-negligible leakage power consumption in one multicore, aggressively lowering the frequency will not always reduce energy consumption. There must be some critical frequencies, below which the system energy will increase again.

Let’s consider one island. We cut the island’s execution line into segments through sorting the cores in the non-decreasing order of the workload, where $C_1$ has the least workload. Due to the operating frequency constraint, each segment operates at only one frequency. Suppose that the execution time and frequency of each segment is $t_j$ and $f_j$ respectively, where $1 \leq j \leq N_c$ (we omit the first index $i$ when considering only one island). An example of one island with four cores ($C_1 \sim C_4$) and four segments ($seg_1 \sim seg_4$) is shown in Figure 1. We can see that each segment $seg_j$ begins from the completion of workload on $C_j-1$ and ends when $C_j$’s workload finishes, and executes $WC_j - WC_{j-1}$ cycles. Moreover, each segment contains $N_c - j + 1$ non-idle cores. The dynamic and leakage energy consumption of each segment $seg_j$ are

$$E_j^d = \alpha(N_c - j + 1)(WC_j - WC_{j-1}) f_j$$
$$E_j^l = (WC_j - WC_{j-1}) f_j$$

Thus, the total energy consumption of the island is

$$E_i^t = \sum_{j=1}^{N_c} (E_j^d + E_j^l)$$

When there are no other constraints, the necessary condition for minimizing the total energy consumption $E_i^t$ is that the gradient of $E_i^t$ is equal to zero, i.e.,

$$\nabla(f_1, ..., f_{N_c}) E_i^t = 0$$

or $\forall j \in [1, N_c], \ \frac{\partial E_i^t}{\partial f_j} = 0$

By solving all equations in (3), we have

$$f_j^* = \sqrt{\frac{P_i}{2\alpha(N_c - j + 1)}}$$

We call these frequencies $\{f_1^*, ..., f_{N_c}^*\}$ critical speed sequence.

If $f_j^* > f_{max}$ ($f_j^* < f_{min}$), we let $f_j^* = f_{max}$ ($f_j^* = f_{min}$).
sequence, and the corresponding finish time/makespan of the workload on the island, i.e., the finish time of the core $C_{N_c}$ with the largest workload, equals to $\sum_{j=1}^{N_c} \frac{W_{C_j} - W_{C_{j-1}}}{f_j}$. Therefore, we have the following observations for a system with no timeout constraint: (i) The optimal frequency of each segment in one island is different for one cluster-based multicore, while there is only one critical speed for per-core DVFS due to each core forming an island. So the critical speed sequence can be seen as a generalization. (ii) The critical speed sequence is not dependent on the workload mapped but the leakage power consumption and the number of cores on an island, so every island has fixed and common critical speed sequence under a symmetric cluster-based multicore system.

IV. ENERGY MINIMIZATION FOR A GIVEN TASK PARTITION

Since the islands are independent with each other in power management, minimizing the energy consumption of each island will also result in global optimal solution. In this section, we consider one island energy minimization problem under a given task partition. The task partition and overall approach are then given in the next section.

When taking account of the deadline constraint, the critical speed sequence may be no longer optimal. We can formulate a constrained convex programming problem\(^3\) to minimize the energy consumption of each island:

$$\begin{align*}
\text{min} & \quad E^t = \sum_{j=1}^{N_c} E_j(t_j) \\
\text{subject to} & \quad \sum_{j=1}^{N_c} t_j \leq D, \\
& \quad t_j^{\text{min}} \leq t_j \leq t_j^{\text{max}}.
\end{align*}$$

(5)

Where $E_j(t_j)$ is the energy consumption of $\text{seg}_j$ which equals to $\frac{\alpha(N_c-j+1)(W_{C_j} - W_{C_{j-1}})^3}{f_j^2} + t_j P_j$, and $t_j^{\text{min}} = \frac{W_{C_j} - W_{C_{j-1}}}{f_j^{\text{max}}}$ and $t_j^{\text{max}} = W_{C_j} - W_{C_{j-1}}$.

We will solve the convex program in two steps. First, narrow the $t_j$'s domain without missing optimality. Second, an algorithm based on the binary search is adopted to derive the optimal solution.

Not the whole domain of $t_j$ is necessary to determine the optimal solution in (5). Since $E_j(t_j)$ is a convex function, there is only one valley point $t_j^*$ which is obtained by setting the first derivative $E_j'(t_j)$ to zero and equals to $t_j^* = \frac{W_{C_j} - W_{C_{j-1}}}{f_j}$. According to the relation between $t_j^*$ and $[t_j^{\text{min}}, t_j^{\text{max}}]$, we narrow the $t_j$'s domain by

$$t_j \in [t_j^{\text{min}}, t_j^{\text{max}}] \text{ if } t_j^{\text{min}} \leq t_j \leq t_j^{\text{max}},$$

(6a)

$$t_j \in [t_j^{\text{min}}, t_j^{\text{max}}] \text{ if } t_j^{\text{min}} < t_j < t_j^{\text{max}}.$$  

(6b)

$$t_j = t_j^{\text{min}} \text{ if } t_j^{\text{min}} \leq t_j \leq t_j^{\text{max}}.$$  

(6c)

If $t_j^*$ is in the range of $[t_j^{\text{min}}, t_j^{\text{max}}]$, we narrow $t_j$'s domain to $[t_j^{\text{min}}, t_j^{\text{max}}]$ in (6a). If $t_j^*$ is greater than or equal to $t_j^{\text{max}}$, the domain remains unchanged in (6b). If $t_j^*$ is less than or equal to $t_j^{\text{min}}$, assign $t_j$ as $t_j^{\text{min}}$ in (6c). Denote $t_j \in [t_j^{\text{low}}, t_j^{\text{up}}]$, where $t_j^{\text{low}}$ and $t_j^{\text{up}}$ are the new lower and upper bound of $t_j$ respectively after narrowing the domain using (6). Then, the second constraint in convex program (5) is reduced to $t_j^{\text{low}} \leq t_j \leq t_j^{\text{up}}$.

Theorem 1. Convex program (5) will miss no optimal solution in the narrowed domain $[t_j^{\text{low}}, t_j^{\text{up}}]$.

Proof: The proof is in our technique report version.

In the narrowed domain $t_j \in [t_j^{\text{low}}, t_j^{\text{up}}]$, $E_j(t_j)$ becomes a monotonotically decreasing function. Hence, when we solve (5) in the case of $\sum_{j=1}^{N_c} t_j^{\text{low}} \leq D < \sum_{j=1}^{N_c} t_j^{\text{up}}$, it must be $\sum_{j=1}^{N_c} t_j = D$ when $E^t$ is minimized. That’s the reason we narrow $t_j$’s domain.

A. An Optimal Algorithm

We propose a polynomial-time complexity algorithm based on the binary search to solve the convex program (5) in the narrowed domain, and then prove its optimality. See Algorithm 1.

Initially Algorithm 1 checks two trivial cases, where the sum of $t_j$’s upper bound is less than $D$ and the sum of $t_j$’s lower bound is greater than $D$ in lines (2) through (7). Lines (8-29) deal with the case when $\sum_{j=1}^{N_c} t_j^{\text{low}} \leq D < \sum_{j=1}^{N_c} t_j^{\text{up}}$, where $\sum_{j=1}^{N_c} t_j = D$ when $E^t$ is minimized. Line (8) sorts the upper and lower bounds of all $E_j(t_j)$ values in decreasing order. The while loop from line (10) to (26) is a binary search which reduces the search space by half in each iteration. In each iteration, all $t_j$ values are derived accordingly in lines (12) through (20). Then, lines from (21) to (25) determine to drop which half of the current interval of $[\text{left}, \text{right}]$. If the sum of current $t_j$ is greater than $D$, i.e., some $t_j$ cannot be assigned as $t_j^{\text{up}}$ but smaller values to meet deadline, then drop the left half by $\text{left} \leftarrow \text{mid}$. If the sum is less than or equal to $D$, i.e., some $t_j$ should not be assigned as $t_j^{\text{low}}$ but larger values to further reduce energy, then drop the right half $\text{right} \leftarrow \text{mid}$. The binary search finally finds an interval $[\text{left}, \text{right}]$, where $t_j = t_j^{\text{low}}$ if $E_j(t_j^{\text{low}}) \geq E_j(t_j^{\text{up}})$ and the rest $t_j$ is less than $t_j^{\text{low}}$. Then, the for loop in lines (27-29) determines the remaining unknown $t_j$ by the Lagrange Multiplier Method. Line (30) returns each segment’s execution time and the island’s energy consumption.

The number of iterations of while loop in line(10) is $O(\log N_c)$ due to the binary search. The iterations of for loop in line (12) and (27) are both at most $O(N_c)$. The complexity of Algorithm 1 is $O(N_c \log N_c)$. Therefore, the complexity is $O(N_c \log N_c)$ when minimizing energy of all $N_c$ islands.

Theorem 2. Algorithm 1 finds the optimal operating frequency sequence leading to the minimized energy consumption for one island with a given task partition.

Proof: The proof is in our technique report version.

V. ENERGY MINIMIZATION FOR DVFS WITH ANY GRANULARITY

In this section, we consider three degrees of freedom including the number of active islands, task partition and frequency...
assignment all together. We first give an example to illustrate the effect of the number of active numbers on the system energy consumption, and then present the overall algorithm which can adopt to DVFS policies with any granularity.

Due to the operating frequency constraint and non-negligible leakage power in each island, mapping a taskset on all islands, i.e., all islands being active, will not always result in a reduced energy consumption. We can see an example shown in Figure 2. When \( D = 12 \), the taskset is partitioned onto two active islands and the schedule is presented as the gray rectangles in Figure 2(a), where each island operates at two different frequencies and the energy consumption is 1.25. If only one island is active, the schedule is shown Figure 2(b), where the island operates at only one frequency and the energy consumption is reduced to 1.09. So, only one island being active results in the minimum energy consumption in this case. When \( D = 3 \), both two islands should be active in order to meet the deadline, which is shown Figure 2(c). The numbers of active islands are different in the two cases.

**Algorithm 1 Binary Search (BS)**

**Input:** \((D, N_c, W C_j)\);

**Output:** \((t_j, E_{ij}^f)\);

1. narrow the domain of \( t_j \);
2. if \( \sum_{j=1}^{N_c} t_j^{up} \leq D \) then
   3. return \( t_j \leftarrow t_j^{up} \);
4. end if
5. if \( \sum_{j=1}^{N_c} t_j^{low} > D \) then
   6. return no solution;
7. end if
8. sort all \( E_j(t_j^{low}) \) and \( E_j(t_j^{up}) \) values in the decreasing order, and denoted as \( \tilde{E}' = \{\tilde{E}_1', \ldots, \tilde{E}_2N_c'\} \);
9. left \( \leftarrow 1 \), right \( \leftarrow 2N_c \);
10. while left \( < \) right \(-1 \) do
11. mid \( \leftarrow \frac{\text{left} + \text{right}}{2} \);
12. for \( j = 1 \) to \( N_c \) do
13. if \( E_{mid}' \leq E_j(t_j^{low}) \) then
14. \( t_j \leftarrow t_j^{low} \);
15. else if \( E_{mid}' \geq E_j(t_j^{up}) \) then
16. \( t_j \leftarrow t_j^{up} \);
17. else
18. \( t_j \leftarrow t \), where \( E_j(t) = \tilde{E}_{mid}' \);
19. end if
20. end for
21. if \( \sum_{j=1}^{N_c} t_j > D \) then
22. left \( \leftarrow \text{mid} \);
23. else \( \{\sum_{j=1}^{N_c} t_j \leq D\} \)
24. right \( \leftarrow \text{mid} \);
25. end if
26. end while
27. for \( j \) such that \( E_j(t_j^{low}) \leq \tilde{E}_{right}' \) and \( E_j(t_j^{up}) \geq \tilde{E}_{left}' \) do
28. determine these \( t_j \) by Lagrange Multiplier Method;
29. end for
30. return \((t_j, E_{ij}^f)\);

**Algorithm 2 Task Partition (LTF)**

**Input:** \((T, D, n_b, N_c)\);

**Output:** \((T_{ij}, WC_{ij})\);

1. sort \( T \) in a non-increasing order of \( wc_n \), where \( \tau_1 \) has the largest WCEC;
2. for \( n = 1 \) to \( N \) do
3. find core \( C_{ij} \) with the smallest workload among all \( n_b \) islands; (break ties by choosing the smallest index \( i \), then smallest \( j \))
4. \( T_{ij} \leftarrow T_{ij} \cup \tau_n \); \( WC_{ij} \leftarrow WC_{ij} + wc_n \);
5. if \( WC_{ij} > D \) then
6. return no solution;
7. end if
8. end for
9. return \((T_{ij}, WC_{ij})\);

Therefore, in order to minimize the energy consumption, we have to find not only the task partition and frequency assignment but also the proper number of active islands. We present the overall algorithm as follows: (i) We first determine the lower bound of the number of islands required to complete the taskset before deadline \( D \), which equals to \( n_{lb} = \left\lceil \frac{\sum_{i=1}^{N_c} wc_i}{N_c D} \right\rceil \), and the upper bound equals to \( n_{ub} = \min\left(\frac{N_c}{\sum_{i=1}^{N_c} wc_i}, N_b\right) \). (ii) A linear search is performed in the interval \([n_{lb}, n_{ub}]\) to determine the proper number of active islands. In each iteration or for each \( n_b \in [n_{lb}, n_{ub}] \), we first adopt the largest task first (LTF) heuristic shown in Algorithm 2 to partition one
taskset onto the \( n_b \) islands employed. Then, Algorithm 1 is used to determine the local minimal energy consumption for the partition of this iteration. (iii) The overall algorithm finally returns the task schedule including the number of active islands, task partition and frequency assignment, which results in the minimum energy value among all of the \((n_{up}^{\text{sp}} - n_{low}^{\text{sp}})\) iterations.

Note that we use LTF heuristic for task partition, so this overall algorithm may not derive the global optimal solution. Recall that the time complexity is \( O(N_b N_c \log N_c) \) when adopting Algorithm 1 to all \( N_b \) islands. It’s lower than the complexity of LTF \( O(N \log N) \) when the taskset size is larger than the number of cores in one multicore. Since there are at most \( N_b \) iterations in the linear search, the time complexity of the overall algorithm is \( O(N_b N \log N) \).

VI. Simulation Results

In this section, we evaluate the energy efficiency of our algorithm proposed in this paper under different cluster partitions. Since no comparison work has addressed the problem defined in this paper from all the three degrees of freedom: the number of active islands, task partition and frequency assignment, for comparison, we have to combine the different approaches for each degree of freedom. We consider two approaches in determining the number of islands employed when task partition: (i) LS. The linear search performed in the overall algorithm in Section V. (ii) AE. The \( N_b \) islands are all employed as the most previous work did, such as [4], [16]. Note that AE is not to have all islands active but consider all islands when task partition, and the number of active islands is determined according to the size of one taskset. If the taskset is so small that each core can not have at least one task, some island may be still inactive. And, two approaches for frequency assignment: (i) BS. The binary search of Algorithm 1 in Section IV. (ii) UF. Aggressively lower the frequency of one island according to the largest workload \( WC_{N_c} \), and have core \( C_{N_c} \) stretch to the deadline. Every island always operates at an unique frequency which equals to \( \max(f_{\text{min}}, \frac{W C_{N_c}}{P_{\text{up}}}) \). We always use Algorithm 2 for task partition to the different combinations of approaches above.

In our simulation, we consider multicore systems with 8, 16 and 32 cores respectively. For the parameters in the power model, it is assumed that \( \alpha = 1 \) and the leakage power consumption \( P_l = 0.1 \times N_c \). Moreover, we assume that \( f_{\text{min}} = 0.01 \), \( f_{\text{max}} = 1 \). The deadline \( D \) of the real-time application is set as 100 time units, and the worst case execution cycle of each task is generated uniformly in the range of \([0.01 \times D, 0.5 \times D]\). The simulations are performed on a Windows PC with an Intel Core2 2.83GHZ 32-bit processor and 2GB main memory. We plot the results averaged over 500 simulation runs.

A. Comparison: Energy Minimization

In this experiment, we evaluate the energy efficiency of several solutions which are combinations of the four approaches above. Figure 3 shows the comparison on energy consumption of LS+BS, AE+BS, AE+UF for the four different cluster partitions of one 32 cores multicore. (Similar results can be obtained in the case of 8 or 16 cores multicore.) The energy consumptions in each figure are normalized to that of the AE+UF. The taskset size ranges from 1 to 64.

LS+BS, i.e., our approach proposed in this paper, significantly outperforms AE+UF across all cluster partitions. AE+UF is a step curve and has \( N_b - 1 \) steps. It “jumps” at every point when the number of tasks equals to the integral multiple of \( N_c \), until the taskset has more than \((N_b - 1) \times N_c\) tasks when all islands become active or have workload mapped on. These integers also reflect the amount of active islands in each step, i.e., the curve also reflects the variance of the active islands with taskset size.

In order to evaluate the effect of the number of active islands on the energy consumption, we compare the results of LS+BS and AE+BS. Except two extreme cases: 1-VFI and 32-VFI cluster partitions which are also known as full-chip and per-core DVFS, LS+BS performs better than AE+BS. LS+BS saves more energy compared to AE+BS up to 16.4% and 11.6% under 2-VFI in Figure 3(a) and 4-VFI cluster partition in Figure 3(b) respectively, and the energy saving decreases as the number of islands \( N_b \) increases. The reason is: since \( N_c \) and \( P_l \) become smaller as \( N_b \) grows, the proportion of one island’s energy consumption to the total energy consumption decreases; even if the amount of active islands in AE and LS are a little different under a cluster partition with fine granularity, their energy consumption will be relatively close.

LS+BS moves towards to AE+BS as the size of taskset grows in Figure 3(a)(b)(c)(d), since the number of active islands increases in LS+BS and will equal to \( N_b \) when taskset size is greater than 40. Besides this reason, the static slack time which equals to \( D - WC_{ij} \) in each \( C_{ij} \) decreases and the space for regulating frequency shrinks as taskset size grows, so LS+BS also moves towards to AE+UF. In sum, the three curves move towards to one another. In general, many systems
This experiment evaluates the energy efficiency for different cluster partitions in 8, 16, 32 and 128 cores multicore through using LS+BS. The energy consumption in each figure is normalized to that of its own 1-VFI cluster partition. The taskset sizes of Figure 4(a)(b)(c) range from 1 to 16, 1 to 32, 1 to 64 and 1 to 128, respectively.

As shown in Figure 4, we can see that energy efficiency increases as $N_t$ increases, and 1-VFI (or full-chip DVFS) has the worst energy efficiency. Moreover, the energy efficiency of various cluster partitions with fewer number of cores on each island is very approximate. For example, the energy consumptions of 8-VFI, 16-VFI and 32-VFI is nearly the same, as shown in Figure 4(c).

We can see that the results of all cluster partitions move towards one another as taskset size grows in all insets of Figure 4. The reason is that the difference of workload among cores decreases and the workload in the first segment increases, so each island operates at one frequency most of the time and the effect of operating frequency constraint weakens.

### VII. CONCLUSIONS

In this paper, we addressed the energy-efficient scheduling problem for real-time tasks on cluster-based multicore systems with non-negligible leakage power consumption, where the proposed algorithm can be adopted into DVFS policies with any granularity. We first proposed a conception of critical speed sequence, and shown that the operating frequencies deriving minimum energy consumption for each island is only dependent on the number of cores and leakage power but not the workload mapped on the island when not considering timing constraint. Second, we presented an optimal polynomial-time complexity algorithm based on binary search to minimize the energy minimization for a real-time multicore system with a fixed task partition. Third, we provided an efficient overall algorithm to determine the proper number of active islands, task partition and frequency assignment, which can avoid to set needless islands on. The simulation results indicated that the proposed algorithm significantly outperforms the related approaches. For future work, we will develop the dynamic slack reclamation algorithms which allow task remapping within one island or across different islands.

### Acknowledgement:
We would like to thank the anonymous reviewers for their constructive comments. This work is partially supported by the NSF of China under Grant No. 60973017 and the Fundamental Research Funds for the Central Universities under Grant No. N100604010 and N100204001.

### REFERENCES