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 multicores (chip-multiprocessors). Although the energy-efficient scheduling problem has been extensively explored for uni-processor and multiprocessor real-time systems [1], little work has addressed multicore systems. Most previous research reduces the energy-efficient scheduling problem for multicores 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 multicores 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.

978-3-9810801-7-9/DATE11/©2011 EDAA

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 4×4 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.

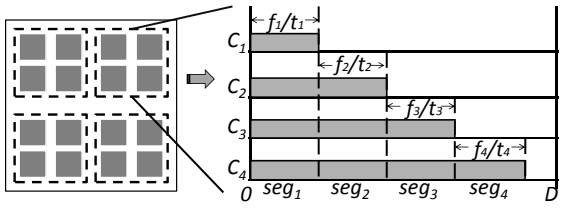


Fig. 1. 4-VFI symmetric partitions for multicores with 16 cores.

Each island has two operational modes [18]: *on* and *off*. We assume the operating frequency can be regulated continuously between f^{min} and f^{max} in an island set on. Moreover, we assume that the frequency values are normalized with respect to the processor's maximum frequency and $f^{max} = 1$. The j^{th} core in i^{th} island B_i is denoted as C_{ij}. The workload of each core is denoted as the worst case execution cycles (WCEC) WC_{ij}, and sorted in a non-decreasing order, i.e., $WC_{ij} \leq WC_{i(j+1)}$. Each workload $WC_{ij} = \sum_{\tau_k \in T_{ij}} wc_k$ and the corresponding execution time is $\frac{WC_{ij}}{f}$ at frequency f, where T_{ij} denotes the task subset executing on core C_{ij} and wc_k denotes the WCEC of task τ_k. We call an island *active*(*inactive*) if the island has (no) workload mapped on.

Since the individual cores of a multicore are still based on the same design as previous single-core processor architectures, the basic principles for leveraging more efficient operation are likely to remain unchanged. We focus on the manageable power consumption of each island, which consists of two parts: dynamic and leakage power. One cores's dynamic power consumption can be expressed in terms of the operating frequency f: $\alpha f^\beta, 2 \leq \beta \leq 3$. We assume $\beta = 3$ in this paper, but the proposed algorithms can be adopted to any convex power function. The dynamic power consumption P^d of an island is the sum of that of each *on* core in the island. We assume that the leakage power consumption P^l of an island is constant, which can be eliminated only when all cores on the island are idle and the island is set off [18]. The total power consumption of the multicore is equal to sum of power consumptions of each island: $\sum_{i=1}^{N_b} (P_i^d + P_i^l)$.

B. Problem Statement

The energy-efficient scheduling problem for cluster-based multicores can be seen as the generalization of that for per-core or full-chip DVFS systems. Specially, each processor can 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

$$\begin{aligned} E_j^d &= \alpha(N_c - j + 1)(WC_j - WC_{j-1})f_j^2 \\ E_j^l &= \tfrac{(WC_j - WC_{j-1})P^l}{f_j} \end{aligned} \tag{1}$$

Thus, the total energy consumption of the island is

$$E^t = \textstyle\sum_{j=1}^{N_c} (E_j^d + E_j^l) \tag{2}$$

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

$$\begin{aligned} &\nabla_{\{f_1,...,f_{N_c}\}} E^t = 0, \\ &or\ \forall j \in [1, N_c],\ \tfrac{\partial E^t}{\partial f_j} = 0 \end{aligned} \tag{3}$$

By solving all equations in (3), we have

$$f_j^* = \sqrt[3]{\tfrac{P^l}{2\alpha(N_c - j + 1)}} \tag{4}$$

We call these frequencies $\{f_1^*, ..., f_{N_c}^*\}$[1] *critical speed se-*

[1]If $f_j^* > f^{max}$ ($f_j^* < f^{min}$), we let $f_j^* = f^{max}$ ($f_j^* = f^{min}$).

quence, 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 $\sum_{j=1}^{N_c} \frac{WC_j - WC_{j-1}}{f_j^*}$. Therefore, we have the following observations for a system with no timing 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[2] 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:

$$\begin{array}{ll} min & E^t = \sum_{j=1}^{N_c} E_j(t_j) \\ sub.to & \sum_{j=1}^{N_c} t_j \le D, \\ & t_j^{min} \le t_j \le t_j^{max}. \end{array} \quad (5)$$

Where $E_j(t_j)$ is the energy consumption of seg_j which equals to $\frac{\alpha(N_c - j + 1)(WC_j - WC_{j-1})^3}{t_j^2} + t_j P^l$, and $t_j^{min} = \frac{WC_j - WC_{j-1}}{f^{max}}$ and $t_j^{max} = \frac{WC_j - WC_{j-1}}{f^{min}}$.

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 $t_j^* = \frac{WC_j - WC_{j-1}}{f_j^*}$. According to the relation between t_j^* and $[t_j^{min}, t_j^{max}]$, we narrow the t_j's domain by

$$\begin{cases} t_j \in [t_j^{min}, t_j^*] \quad if\ t_j^{min} < t_j^* < t_j^{max} & (6a) \\ t_j \in [t_j^{min}, t_j^{max}] \quad if\ t_j^{max} \le t_j^* & (6b) \\ t_j = t_j^{min} \quad if\ t_j^* \le t_j^{min} & (6c) \end{cases}$$

If t_j^* is in the range of $[t_j^{min}, t_j^{max}]$, we narrow t_j's domain to $[t_j^{min}, t_j^*]$ in 6(a). If t_j^* is greater than or equal to t_j^{max}, the domain remains unchanged in 6(b). If t_j^* is less than or equal to t_j^{min}, assign t_j as t_j^{min} in 6(c). Denote $t_j \in [t_j^{low}, t_j^{up}]$, where t_j^{low} and t_j^{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 $t_j^{low} \le t_j \le t_j^{up}$.

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

Proof: The proof is in our technique report version. ■

In the narrowed domain $t_j \in [t_j^{low}, t_j^{up}]$, $E_j(t_j)$ becomes a monotonously decreasing function. Hence, when we solve (5) in the case of $\sum_{j=1}^{N_c} t_j^{low} \le D < \sum_{j=1}^{N_c} t_j^{up}$, 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 $\sum_{j=1}^{N_c} t_j^{low} \le D < \sum_{j=1}^{N_c} t_j^{up}$, 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 $[left, right]$. If the sum of current t_j is greater than D, i.e., some t_j can not be assigned as t_j^{up} but smaller values to meet deadline, then drop the left half by $left \Leftarrow mid$. If the sum is less than or equal to D, i.e., some t_j should not be assigned as t_j^{low} but larger values to further reduce energy, then drop the right half $right \Leftarrow mid$. The binary search finally finds an interval $[left, right]$, where $t_j = t_j^{low}$ if $E_j'(t_j^{low}) \ge \widehat{E}'_{left}$, $t_j = t_j^{up}$ if $E_j'(t_j^{up}) \le \widehat{E}'_{right}$ and the rest t_j is greater than t_j^{low} and less than t_j^{up}. 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 $O(logN_c)$ 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 $O(N_c logN_c)$. Therefore, the complexity is $O(N_b N_c logN_c)$ when minimizing energy of all N_b 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.*

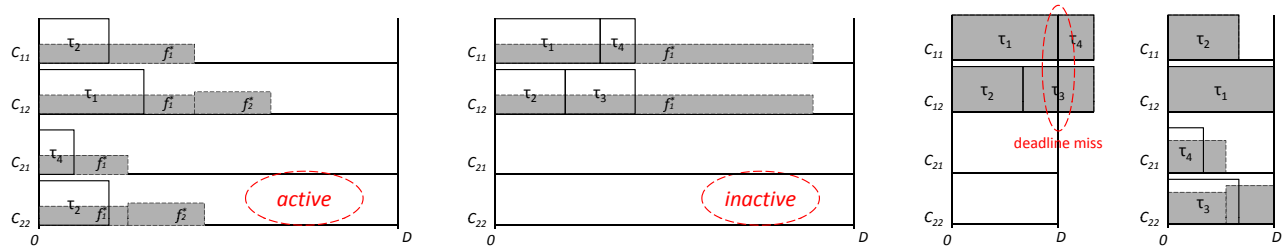
Proof: The proof is in our technique report version. ■

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

[2]This number, i.e., N_c in (4), becomes the number of cores with workload mapped in one island, if some core has no workload mapped.

[3]It's easy to prove that E^t in (5) is a convex function by the proof of the convexity of each $E(t_j)$. Moreover, all constraints are linear inequations.



(a) 2 active islands, D=12 (b) 1 active island, D=12 (c) D=3

Fig. 2. Different task schedules of four tasks in a 2 × 2 cluster-based multicore system. The system parameters are shown in Table I. The widths of the rectangles denote the execution time while their heights represent the frequency.

Algorithm 1 Binary Search (BS)

Input: (D, N_c, WC_j);
Output: (t_j, E_i^t);
narrow the domain of t_j;
if $\sum_{j=1}^{N_c} t_j^{up} \leq D$ **then**
  return $t_j \Leftarrow t_j^{up}$;
end if
if $\sum_{j=1}^{N_c} t_j^{low} > D$ **then**
  return no solution;
end if
sort all $E'_j(t_j^{low})$ and $E'_j(t_j^{up})$ values in the decreasing order, and denoted as $\widehat{E}' \Leftarrow \{\widehat{E}'_1, ..., \widehat{E}'_{2N_c}\}$;
$left \Leftarrow 1$, $right \Leftarrow 2N_c$;
while $left < right - 1$ **do**
  $mid \Leftarrow \left\lfloor \frac{left+right}{2} \right\rfloor$;
  for $j = 1$ to N_c **do**
    if $\widehat{E}'_{mid} \leq E'_j(t_j^{low})$ **then**
      $t_j \Leftarrow t_j^{low}$;
    else if $\widehat{E}'_{mid} \geq E'_j(t_j^{up})$ **then**
      $t_j \Leftarrow t_j^{up}$;
    else
      $t_j \Leftarrow t$, where $E'_j(t) = \widehat{E}'_{mid}$;
    end if
  end for
  if $\sum_{j=1}^{N_c} t_j > D$ **then**
    $left \Leftarrow mid$;
  else $\{\sum_{j=1}^{N_c} t_j \leq D\}$
    $right \Leftarrow mid$;
  end if
end while
for j such that $E'_j(t_j^{low}) \leq \widehat{E}'_{right}$ and $E'_j(t_j^{up}) \geq \widehat{E}'_{left}$ **do**
  determine these t_j by Lagrange Multiplier Method;
end for
return (t_j, E_i^t);

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.

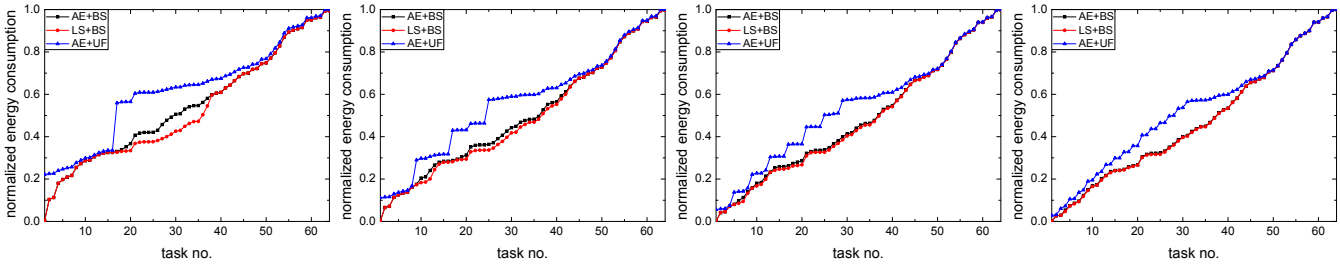
TABLE I
THE SYSTEM PARAMETERS.

α	P^l	wc_1	wc_2	wc_3	wc_4
1	0.2	3	2	2	1

Algorithm 2 Task Partition (LTF)

Input: (T, D, n_b, N_c);
Output: (T_{ij}, WC_{ij});
sort T in a non-increasing order of wc_n, where τ_1 has the largest WCEC;
for $n = 1$ to N **do**
  find core C_{ij} with the smallest workload among all n_b islands; (break ties by choosing the smallest index i, then smallest j)
  $T_{ij} \Leftarrow T_{ij} \cup \tau_n$; $WC_{ij} \Leftarrow WC_{ij} + wc_n$;
  if $WC_{ij} > D$ **then**
    return no solution;
  end if
end for
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 $n_b^{low} = \left\lceil \frac{\sum_{i=1}^{N} wc_i}{N_c D} \right\rceil$, and the upper bound equals to $n_b^{up} = min(\left\lceil \frac{N}{N_c} \right\rceil, N_b)$. (ii) A linear search is performed in the interval $[n_b^{low}, n_b^{up}]$ to determine the proper number of active islands. In each iteration or for each $n_b \in [n_b^{low}, n_b^{up}]$, we first adopt the largest task first (LTF) heuristic shown in Algorithm 2 to partition one



(a) 2-VFI (b) 4-VFI (c) 8-VFI (d) 16-VFI

Fig. 3. Comparison of three solutions: AE+BS, LS+BS and AE+UF. x-VFI: the multicore contains x islands.

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 $(n_b^{up} - n_b^{low})$ 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 $O(NlogN)$ 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 $O(N_b NlogN)$.

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 $max(f^{min}, \frac{WC_{N_c}}{D})$. 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 $P^l = 0.1 * N_c$. Moreover, we assume that $f^{min} = 0.01, f^{max} = 1$. 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 $[0.01 * D, 0.5 * D]$. 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.

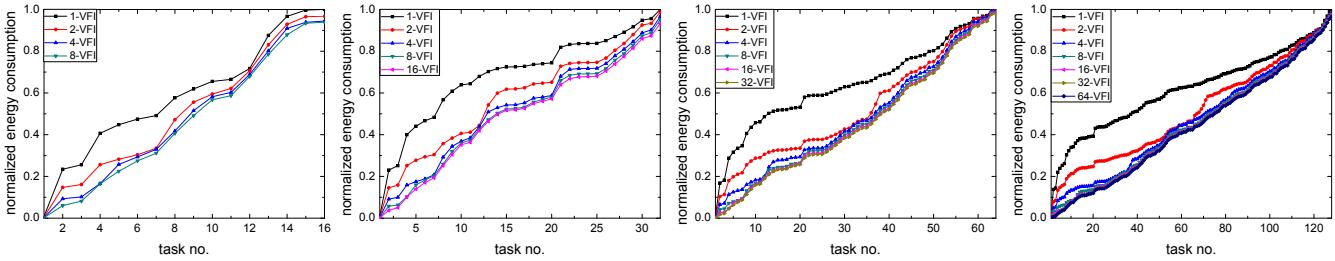
A. Comparison: Energy Minimization

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 $(N_b - 1) * N_c$ 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



(a) 8 cores (b) 16 cores (c) 32 cores (d) 64 cores

Fig. 4. Comparison of energy efficiency for different cluster partitions in 8, 16, 32 and 64 cores multicore.

operate under mid-range loads, hence the energy saving when the task number is between 20 to 40, is very encouraging.

B. Comparison: Cluster Partitions

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_b 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 to 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

[1] J. Chen and C. Kuo, "Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms," in *RTCSA'07*, pp. 28–38.
[2] F. Gruian, "System-Level Design Methods for Low-Energy Architectures Containing Variable Voltage Processors," in *PACS'00*, p. 1.
[3] F. Gruian and K. Kuchcinski, "LEneS: task scheduling for low-energy systems using variable supply voltage processors," in *ASP-DAC'01*, p. 455.
[4] J. Chen, H. Hsu, K. Chuang, C. Yang, A. Pang, and T. Kuo, "Multiprocessor energy-efficient scheduling with task migration considerations," in *ECRTS'04*, pp. 101–108.
[5] J. Chen and T. Kuo, "Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics," in *ICPP'05*, pp. 13–20.
[6] J. Chen, H. Hsu, and T. Kuo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," in *RTAS'06*, pp. 408–417.
[7] J. Chen, T. Kuo, C. Yang, and K. King, "Energy-efficient real-time task scheduling with task rejection," in *DATE'07*, pp. 1–6.
[8] J. Chen, C. Yang, H. Lu, T. Kuo, S. Zurich, and T. Taipei, "Approximation algorithms for multiprocessor energy-efficient scheduling of periodic real-time tasks with uncertain task execution time," in *RTAS'08*, pp. 13–23.
[9] Y. Zhang, X. Hu, and D. Chen, "Task scheduling and voltage selection for energy minimization," in *DAC'02*, pp. 183–188.
[10] R. Mishra, N. Rastogi, D. Zhu, D. Mosse, and R. Melhem, "Energy aware scheduling for distributed real-time systems," in *IPDPS'03*, p. 9.
[11] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *IPDPS'03*, pp. 113–121.
[12] T. AlEnawy and H. Aydin, "Energy-aware task allocation for rate monotonic scheduling," in *RTAS'05*, pp. 213–223.
[13] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé, "Energy-efficient policies for embedded clusters," in *LCTES'05*, pp. 1–10.
[14] C. Xian, Y. Lu, and Z. Li, "Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time," in *DAC'07*, p. 669.
[15] E. Seo, J. Jeong, S. Park, and J. Lee, "Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors," *TPDS*, vol. 19, no. 11, pp. 1540–1552, 2008.
[16] C. Yang, J. Chen, and T. Kuo, "An approximation algorithm for energy-efficient scheduling on a chip multiprocessor," in *DATE'05*, pp. 468–473.
[17] S. Borkar, "Thousand core chips: a technology perspective," in *DAC'07*, pp. 746–749.
[18] D. Dal, A. Nunez, and N. Mansouri, "Power islands: a high-level technique for counteracting leakage in deep sub-micron," in *ISQED'06*, pp. 165–170.
[19] A. Das, S. Ozdemir, G. Memik, and A. Choudhary, "Evaluating voltage islands in CMPs under process variations," in *ICCD'07*, pp. 129–136.
[20] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *ISLPED'07*, pp. 38–43.
[21] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu, "Architecting voltage islands in core-based system-on-a-chip designs," in *ISLPED'04*, pp. 180–185.
[22] D. Lackey, P. Zuchowski, T. Bednar, D. Stout, S. Gould, and J. Cohn, "Managing power and performance for System-on-Chip designs using Voltage Islands," in *ICCAD'02*, pp. 195–202.