Minimizing Energy Consumption of Real-Time Parallel Applications Using Downward and Upward Approaches on Heterogeneous Systems

Guoqi Xie, *Member, IEEE*, Junqiang Jiang, Yan Liu, Renfa Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—The problem of minimizing the energy consumption of a real-time parallel application on a heterogeneous system has been studied recently, and slack time reclamation based on the dynamic voltage and frequency scaling (DVFS) energy-efficient design technique has been proposed as a solution. However, the state-of-the-art algorithms merely minimize energy consumption through an "upward" approach (i.e., from exit to entry tasks) and do not apply the "downward" approach (i.e., from entry to exit tasks) to energy consumption minimization. This study solves the same problem by employing "downward" and "upward" approaches. The concepts of deadline-slack and task level are introduced to transfer the deadline of the parallel application to each task, that is, "downward" energy consumption minimization is implemented. "Upward" energy consumption minimization by reclaiming the slack time is then included to implement "downward" and "upward" energy consumption minimization with low time complexity. Results of the experiments using real parallel applications show that the proposed algorithm can generate the minimum energy consumption compared with the state-of-the-art algorithms under different real-time and scale conditions.**

***Index Terms*—Dynamic voltage and frequency scaling (DVFS), energy consumption, heterogeneous systems, real-time.**

Manuscript received May 14, 2016; revised August 17, 2016 and October 12, 2016; accepted February 26, 2017. Date of publication March 1, 2017; date of current version June 1, 2017. This work was supported in part by the National Key Research and Development Plan of China under Grant 2016YFB0200405 and Grant 2012AA01A301-01, in part by the National Natural Science Foundation of China under Grant 61672217, Grant 61173036, Grant 61432005, Grant 61300037, Grant 61300039, Grant 61379115, Grant 61402170, Grant 61370097, Grant 61502162, and Grant 61502405, in part by the CERNET Innovation Project under Grant NGII20161003, and in part by the China Postdoctoral Science Foundation under Grant 2016M592422. Paper no. TII-16-0449. *(Corresponding author: R. Li).*

G. Xie, J. Jiang, Y. Liu, and R. Li are with the College of Computer Science and Electronic Engineering, and the Key Laboratory for Embedded and Network Computing, Hunan University, Changsha 410082, China (e-mail: xgqman@hnu.edu.cn; jjq@hnu.edu.cn; liuyan@hun.edu.cn; lirenfa@hnu.edu.cn).

K. Li is with the College of Computer Science and Electronic Engineering and the Key Laboratory for Embedded and Network Computing, Hunan University, Changsha, 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2676183

I. Introduction

A. Background

THE ongoing progress in semiconductor technology has allowed for constructing fascinating, complex heterogeneous microprocessors systems [1]–[6]. However, this technological progress has led computing hit a power and complexity wall. Thus, energy efficiency has become the key driver behind performance scaling and energy consumption minimization is one of the primary design requirements on all scales, from portable devices, such as smartphones and tablet PCs, to high-performance computing systems. The popular energy consumption optimization technique dynamic voltage and frequency scaling (DVFS) achieves energy-efficient optimization by simultaneously scaling down both the supply voltage and frequency of the processor while tasks are running [1], [6]–[12].

B. Motivation

The deadline is an important design constraint for real-time applications; missing the deadlines of these applications could cause serious disastrous consequences, such as damage, injury, and even death [13]. Some studies have been recently conducted to minimize energy consumption while satisfying the deadline constraint [14], [15]; however, these studies are restricted to independent tasks. As heterogeneous multiprocessors continue to be scaled up, distributed and parallel applications with precedence-constrained tasks, such as fast Fourier transform and Gaussian elimination applications, are increasing in number on heterogeneous systems [16]. A parallel application with precedence-constrained tasks is represented by a directed acyclic graph (DAG), in which the nodes represent the tasks and the edges represent the communication messages between tasks [3], [16]–[18].

The problem of minimizing the energy consumption of a real-time application with precedence-constrained tasks has been solved recently in a number of studies [9], [19]. However, these studies are merely interested in homogeneous systems. The same problem has been studied for heterogeneous systems by reclaiming the slack time [20], [21] from the exit to the entry tasks based on the latest finish time (LFT); however, this strategy merely minimizes the energy consumption through an "upward" approach (i.e., from exit to entry tasks), and does not apply the "downward" approach (i.e., from entry to exit tasks) to energy consumption minimization. For a distributed parallel application, applying both "downward" and "upward" approaches to energy consumption minimization may be more efficient.

C. Main Contributions

The contributions of this study are summarized as follows.

1) The "downward" energy consumption minimization (DECM) algorithm is proposed. "Downward" means that energy consumption minimization is implemented from the entry to the exit tasks according to the descending order of rank upward values (rank_u). In DECM, the concepts of deadline-slack and task level are introduced for the parallel application to transfer the deadline for an application to each task. In this way, all the tasks can be finished within individual deadlines without concerning for the deadline of the application, such that a low time complexity can be achieved.

2) The "upward" energy consumption minimization algorithm is integrated into DECM to implement "downward" and "upward" energy consumption minimization (DUECM). "Upward" means that energy consumption minimization is implemented from the exit to the entry tasks according to the descending order of LFTs. DUECM can eliminate or reduce the slacks between adjacent tasks in the same processor while still satisfying the deadline constraint of the application.

3) Experiments using fast Fourier transform and Gaussian elimination applications are extensively conducted and the results of these experiments consistently verify that the proposed DUECM can generate the minimum energy consumption compared with the state-of-the-art algorithms under different real-time and scale conditions.

II. Related Work

Energy consumption optimization for processors by DVFS has been extensively studied since the mid-1990s. Weiser *et al.* [22] first proposed an approach to optimize energy consumption by using fine-grained control for CPU speed of an operating system scheduler. The aforementioned studies inspired a substantial number of investigations on energy consumption and energy reduction by slack time reclamation, which is an important energy consumption minimization method employed to solve the problem of scheduling independent or precedence-constrained tasks to optimize energy consumption of a uniprocessor or multiprocessors [23], [24].

Low-power and energy-efficient design techniques and algorithms aim at minimizing energy consumption while still meeting certain requirements, especially on deadline constraints. The problems of minimizing the expected energy consumption for a single task with a deadline and randomized execution time for a uniprocessor [25] and homogeneous multiprocessors have been presented [14]. Li [26] studied the power-aware task scheduling of independent sequential tasks on homogeneous multiprocessors with DVFS as combinatorial optimization problems. Li *et al.* [27] studied the problem of scheduling a collection of independent tasks on heterogeneous systems with deadline and energy consumption constraints.

Energy consumption optimization for parallel applications with precedence-constrained tasks has been studied recently. Zong *et al.* [28] considered energy-aware duplication scheduling algorithms for a parallel application on homogeneous systems. Lee and Zomaya [29] presented energy-conscious scheduling to simultaneously minimize the energy consumption and schedule length of a parallel application on heterogeneous systems. However, the aforementioned studies do not consider the deadline constraint of the application. The problem of minimizing the energy consumption of a real-time application with precedence-constrained sequential tasks [9] and precedence-constrained parallel tasks (i.e., a parallel application) [19] has been presented in different studies. However, these studies are merely interested in homogeneous multiprocessors with shared memory (i.e., without communication time between tasks). Huang *et al.* [20] studied the problem of minimizing energy consumption of a real-time parallel application on heterogeneous systems by the enhanced energy-efficient scheduling (EES) algorithm, which reclaims time through the upward approach. Tang *et al.* [21] solved the same problem by presenting the DVFS-enabled energy-efficient workflow task scheduling (DEWTS), which introduces the feature of turning off the relatively inefficient processors to realize the EES-based slack time reclamation [20]. However, in DEWTS, the slack time reclamation by turning off the processors is practically unrealistic and increases the time complexity. Moreover, the state-of-the-art algorithms [20], [21] merely minimize energy consumption through an "upward" approach and do not apply the "downward" approach to energy consumption minimization. This study solves the same problem by simultaneously applying the "downward" and "upward" approaches to implement low time-complexity energy consumption minimization.

III. Models and Preliminaries

A. Application Model

In this study, $U = \{u_1, u_2, \ldots, u_{|U|}\}$ represents a set of heterogeneous processors. A parallel application is represented by the DAG [3], [6], [11], [12], [16]–[18], [20], [21] $G = (N, M, W)$. N represents a set of nodes in G, and each node $n_i \in N$ represents a task with different execution time on different processors. Given that systems with variable processing frequencies are considered in this study, the execution time of task n_i corresponds to the execution time under the maximum processing frequency $f_{\max}$ on a processor [12]. M is a set of communication edges, and each edge $m_{i,j} \in M$ represents the communication message from n_i to n_j. Accordingly, $c_{i,j}$ represents the communication time of $m_{i,j}$. The sets of immediate predecessor and successor tasks of n_i are denoted by $\text{pred}(n_i)$ and $\text{succ}(n_i)$, respectively. The task that has no predecessor task is denoted by n_{entry}; and the task that has no successor task is denoted by n_{exit}. W is an $|N| \times |U|$ matrix where $w_{i,k}$ denotes the execu-

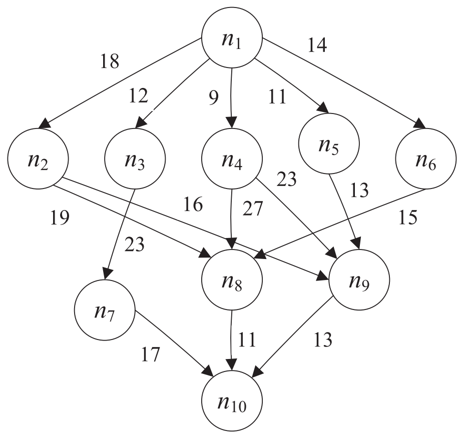


Fig. 1. Standard example of a DAG-based parallel application [16]–[18].

TABLE I
EXECUTION TIME MATRIX IN FIG. 1 [16]–[18]

Task	u_1	u_2	u_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16

tion time of n_i running on the processor u_k with the maximum frequency.

Fig. 1 shows a standard example of a DAG-based parallel application with assigned task and edge weights. This example has been employed in numerous studies [16], [18]. Table I presents a matrix of the execution time for the tasks in Fig. 1. This example shows ten tasks executed on three processors. The weight 14 for n_1 on u_1 in Table I represents the execution time with the maximum frequency and is denoted by $w_{1,1} = 14$. The same task has different execution time on different processors owing to the heterogeneity of the processors. The weight 18 of the edge between n_1 and n_2 represents the communication time denoted by $c_{1,2}$.

The motivating example will be used to explain the proposed methods in this paper. For simplicity, all the units of all parameter values are omitted in the example.

B. Power Model

Given the nearly linear relationship between the supply voltage and operating frequency, DVFS scales down the supply voltage alongside the processing frequency to optimize energy consumption. As in [11] and [12], the term frequency change in this study refers to the coupled change in the supply voltage and processing frequency. Considering a DVFS-capable system, the system-level power model used in [11] and [12] is also adopted in this study. In this model, the system power consumption at frequency f is given by

$$P(f) = P_{\mathrm{s}} + h(P_{\mathrm{ind}} + P_{\mathrm{d}}) = P_{\mathrm{s}} + h(P_{\mathrm{ind}} + C_{\mathrm{ef}} f^m) \quad (1)$$

where P_s represents the static power, which can be removed only by turning off the entire system; P_{ind} represents the frequency-independent dynamic power, which can be removed by setting the systems into the sleep state; P_{d} represents the frequency-dependent dynamic power, which depends on the processing frequencies; and h represents the system state and indicates whether the dynamic powers are currently consumed in the system (when the system is active, $h = 1$; otherwise, $h = 0$); C_{ef} represents the effective switching capacitance; and m represents the dynamic power exponent, which is at least 2. Both C_{ef} and m are processor-dependent constants.

No excessive overhead is associated with the turning ON/OFF of a system; P_{s} is always consumed and unmanageable [11], [12]. As in the aforementioned studies, this study concentrates on managing the dynamic powers (i.e., P_{ind} and P_{d}). Owing to the frequency-independent dynamic power P_{ind}, a lower frequency-dependent dynamic power P_{d} does not result in lower energy consumption. Thus, a minimum energy-efficient frequency f_{ee} exists [11], [12], and it is denoted by

$$f_{\mathrm{ee}} = \sqrt[m]{\frac{P_{\mathrm{ind}}}{(m-1)C_{\mathrm{ef}}}}. \quad (2)$$

If the operating frequency of a processor varies from the minimum available frequency f_{min} to the maximum frequency f_{max}, then the lowest frequency for executing a task should be

$$f_{\mathrm{low}} = \max(f_{\mathrm{min}}, f_{\mathrm{ee}}). \quad (3)$$

Therefore, any actual effective frequency f_h should satisfy the following relation: $f_{\mathrm{low}} \leqslant f_{\mathrm{h}} \leqslant f_{\mathrm{max}}$.

Given that the number of processors is $|U|$ in the system and these processors are completely heterogeneous, each processor should have an individual frequency-independent dynamic power P_{ind}; frequency-dependent dynamic power P_{d}; and an actual effective frequency set. This study defines the frequency-independent dynamic power set as $\{P_{1,\mathrm{ind}}, P_{2,\mathrm{ind}}, \ldots, P_{|U|,\mathrm{ind}}\}$, the frequency-dependent dynamic power set as $\{P_{1,\mathrm{d}}, P_{2,\mathrm{d}}, \ldots, P_{|U|,\mathrm{d}}\}$, the effective switching capacitance set as $\{C_{1,\mathrm{ef}}, C_{2,\mathrm{ef}}, \ldots, C_{|U|,\mathrm{ef}}\}$, the dynamic power exponent set as $\{m_1, m_2, \ldots, m_{|U|}\}$, the minimum energy-efficient frequency set as $\{f_{1,\mathrm{ee}}, f_{2,\mathrm{ee}}, \ldots, f_{|U|,\mathrm{ee}}\}$, and the actual effective frequency set as

$$\left\{\begin{array}{c} \{f_{1,\mathrm{low}}, f_{1,\alpha}, f_{1,\beta}, \ldots, f_{1,\mathrm{max}}\}, \\ \{f_{2,\mathrm{low}}, f_{2,\alpha}, f_{2,\beta}, \ldots, f_{2,\mathrm{max}}\}, \\ \ldots, \\ \{f_{|U|,\mathrm{low}}, f_{|U|,\alpha}, f_{|U|,\beta}, \ldots, f_{|U|,\mathrm{max}}\} \end{array}\right\}.$$

C. Preliminaries

The heterogeneous earliest finish time (HEFT) algorithm is the most popular DAG-based scheduling algorithm for reducing the schedule length to a minimum while achieving low complexity and high performance in heterogeneous systems [16].

Upward rank value: HEFT uses the upward rank value (rank_u) of a task [see (4)] as the task priority standard. In this case, the

TABLE II
UPWARD RANK VALUE, LOWER BOUNDS, AND DEADLINES FOR TASKS OF THE APPLICATION IN FIG. 1

Task	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$\text{rank}_u(n_i)$	108	77	80	80	69	63.3	42.7	35.7	44.3	14.7
$LB(n_i)$	9	40	28	26	38	42	49	62	68	80
$L(n_i)$	1	2	2	2	2	2	3	3	3	4
$D(n_i)$	14	50	38	36	48	52	64	77	83	100

tasks are ordered according to the descending order of rank_u. Table II shows the upward rank values of all the tasks in Fig. 1; these values are obtained by (4)

$$\text{rank}_u(n_i) = \overline{w_i} + \max_{n_j \in \text{succ}(n_i)} \{c_{i,j} + \text{rank}_u(n_j)\} \quad (4)$$

where $\overline{w_i}$ represents the average execution time of task n_i and calculated by $\overline{w_i} = (\sum_{k=1}^{|U|} w_{i,k})/|U|$.

Table II shows the upward rank values of all the tasks of the application in Fig. 1 and the task assignment order in G is $\{n_1, n_3, n_4, n_2, n_5, n_6, n_9, n_7, n_8, n_{10}\}$.

Earliest finish time: The attributes $\text{EST}(n_i, u_k, f_{k,\max})$ and $\text{EFT}(n_i, u_k, f_{k,\max})$ represent the earliest start time (EST) and earliest finish time (EFT), respectively, of task n_i on the processor u_k, with the maximum frequency $f_{k,\max}$. $\text{EFT}(n_i, u_k, f_{k,\max})$ is considered the task assignment criterion because it can meet the local optimum of each task. The aforementioned attributes are calculated by

$$\begin{cases} \text{EST}(n_{\text{entry}}, u_k, f_{k,\max}) = 0 \\ \text{EST}(n_i, u_k, f_{k,\max}) = \max\left\{ \text{avail}[k], \max\limits_{n_x \in \text{pred}(n_i)} \{\text{AFT}(n_x) + c'_{x,i}\}\right\} \end{cases} \quad (5)$$

and

$$\text{EFT}(n_i, u_k) = \text{EST}(n_i, u_k) + w_{i,k}. \quad (6)$$

In (6), avail$[k]$ is the earliest available time when processor u_k is ready for task execution, $\text{AFT}(n_x)$ is the actual finish time (AFT) of task n_x, and $c'_{x,i}$ represents the communication time between n_x and n_i. If n_x and n_i are assigned to the same processor, then $c'_{x,i} = 0$; otherwise, $c'_{x,i} = c_{x,i}$.

Lower bound and deadline: Similar to the state of the art [13], [20], [21], this study also employees HEFT to assess the lower bound of a parallel application. The lower bound, which refers to the minimum schedule length of an application when all the processors are monopolized by the application by using the standard DAG-based scheduling algorithm, is calculated by

$$LB(G) = \min_{u_k \in U} \{\text{EFT}(n_{\text{exit}}, u_k, f_{k,\max})\}. \quad (7)$$

Fig. 2 shows the HEFT-based task assignment and scheduling of the parallel application G in Fig. 1, where $LB(G) = 80$. The arrows in Fig. 2 represent the generated communication time on different processors. A relative deadline (i.e., $D(G)$) for the application is provided based on the actual physical time requirement after a hazard analysis and risk assessment.

Maximum energy consumption: Given that the maximum frequency is used to calculate the EFT of each task on each processor, HEFT is employed to assess the maximum energy consumption of a parallel application in this study. Calculated

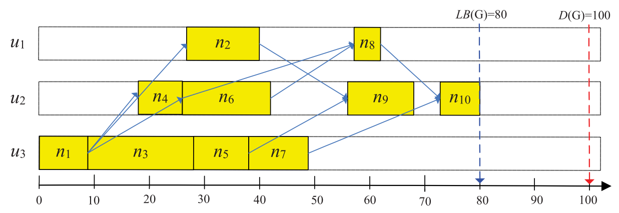


Fig. 2. HEFT-generated scheduling of the application in Fig. 1.

by (8), the maximum energy consumption is reached when the lower bound is obtained

$$\begin{aligned} E_{\max}(G) &= \sum_{i=1}^{|N|} E\left(n_i, u_{pr(i)}, f_{pr(i),\max}\right) \\ &= \sum_{i=1}^{|N|} \left(P_{pr(i),\text{ind}} + C_{pr(i),\text{ef}} \times (f_{pr(i),\max})^{m_{pr(i)}}\right) \\ &\quad \times w_{i,pr(i)} \end{aligned} \quad (8)$$

where $E(n_i, u_k, f_{pr(i),\max})$ represents the energy consumption of the task n_i assigned to the processor $u_{pr(i)}$ with the maximum frequency $f_{pr(i),\max}$. The energy consumption values of the application G is $E_{\max}(G) = 159.49$.

IV. ENERGY CONSUMPTION MINIMIZATION

A. Problem Description

Let $E(n_i, u_k, f_{k,h})$ represent the energy consumption of n_i on the processor u_k with frequency $f_{k,h}$ and be calculated by

$$E(n_i, u_k, f_{k,h}) = (P_{k,\text{ind}} + C_{k,\text{ef}} \times (f_{k,h})^{m_k}) \times \frac{w_{i,k} \times f_{k,\max}}{f_{k,h}}. \quad (9)$$

The schedule length of the application G is calculated by

$$SL(G) = \min_{u_k \in U} \left\{ \min_{f_{k,h} \in [f_{k,\text{low}}, f_{k,\max}]} \{\text{EFT}(n_{\text{exit}}, u_k, f_{k,h})\} \right\}. \quad (10)$$

Suppose a distributed system with a parallel application and heterogeneous processors, which support different frequency levels. Then, the problem to be addressed in this study is to assign an available processor with a proper frequency to each task while minimizing energy consumption and ensuring that the generated schedule length of the application is safe (i.e., not exceeding the deadline). The formal description is to find the processor and frequency assignments for all the tasks and to minimize energy consumption

$$E(G) = \sum_{i=1}^{|N|} E(n_i) = \sum_{i=1}^{|N|} E\left(n_i, u_{pr(i)}, f_{pr(i),hz(i)}\right) \quad (11)$$

where $u_{pr(i)}$ and $f_{pr(i),hz(i)}$ represent the assigned processor and frequency of n_i, respectively, subject to

$$SL(G) \leqslant D(G)$$

and

$$f_{pr(i),\text{low}} \leqslant f_{pr(i),hz(i)} \leqslant f_{pr(i),\max}$$

for $\forall i : 1 \leqslant i \leqslant |N|, u_{pr(i)} \in U$.

B. Deadline Slack

Definition 1: (*Deadline slack):* The deadline slack of an application represents the value of the deadline minus the lower bound of the application, that is,

$$DS(G) = D(G) - LB(G)\,. \tag{12}$$

Given that the HEFT algorithm is used to calculate $LB(G)$, then $DS(G)$ is actually determined by $D(G)$. In this study, the deadline of the application is divided into tasks according to their task levels. The task level of each task is calculated by

$$\begin{cases} L(n_{\text{entry}}) = 1 \\ L(n_i) = \max\limits_{n_x \in \text{pred}(n_i)} \{L(n_x) + 1\} \end{cases}. \tag{13}$$

In this study, the deadline slacks of all the tasks are equal to the deadline slack of the application, i.e., $DS(n_i) = DS(G)$. Thereafter, the deadline of task n_i ($n_i \in N$) can be generated. Thus,

$$D(n_i) = LB(n_i) + \tfrac{DS(G)}{L(n_{\text{exit}})} \times L(n_i) \tag{14}$$

where $LB(n_i)$ represents the lower bound of n_i, which is equal to AFT(n_i) in the HEFT algorithm. That is, all tasks have individual lower bounds. For example, when $D(G) = 100$, the deadlines of all the tasks in Fig. 1 are those shown in Table II.

The HEFT algorithm executes all the tasks with the maximum frequencies and minimum EFT on all the processors, as mentioned in Section III-C. After defining a deadline for each task [see (14)], a proper processor $u_{pr(i)}$ and corresponding frequency $f_{pr(i),hz(i)}$ for the task n_i that minimize the energy consumption for each task should be determined as follows:

$$E(n_i) = \min_{u_k \in U} \left\{ \min_{f_{k,h} \in [f_{k,\text{low}},\, f_{k,\max}]} \{E(n_i, u_k, f_{k,h})\} \right\}$$

subject to

$$\text{EFT}(n_i, u_k, f_{k,h}) \leqslant D(n_i)$$

where

$$\text{EFT}(n_i, u_k, f_{k,h}) = \text{EST}(n_i, u_k, f_{k,h}) + w_{i,k} \times \frac{f_{k,\max}}{f_{k,h}} \tag{15}$$

and

$$f_{pr(i),\text{low}} \leqslant f_{pr(i),hz(i)} \leqslant f_{pr(i),\max}$$

for $\forall i : 1 \leqslant i \leqslant |N|, u_{pr(i)} \in U$. Consequently, the energy consumption problem of the application is transferred to that of each task. The actual start time (AST) of n_i on the processor $u_{pr(i)}$ with frequency $f_{pr(i),hz(i)}$ should be

$$\text{AST}(n_i) = \text{AFT}(n_i) - f_{pr(i),\max} \times \frac{w_{i,pr(i)}}{f_{pr(i),hz(i)}}. \tag{16}$$

C. DECM Algorithm

The DECM algorithm to minimize energy consumption is presented in this section. The steps of DECM are provided in Algorithm 1.

In DECM, all the processors for all the tasks are assigned in advance by using the HEFT algorithm; DECM is mainly to scale down the maximum frequency to a lower level for each task on the same processor. That is, each task selects the processor's frequency that generates the minimum energy consumption within its deadline constraint. The core details are explained as follows.

Algorithm 1: The DECM Algorithm.

```
Input: $U = \{u_1, u_2, \ldots, u_{|U|}\}, G$
Output: $SL(G)$, $E(G)$
 1: Sort the tasks in a list downward_task_list according to
    the descending order of rank$_u$ values.
 2: Call the HEFT algorithm [16] to obtain the assigned
    processor and AFT of each task and the lower bound
    $LB(G)$ of the application $G$;
 3: Compute $DS(G)$ by (12);
 4: while (there are tasks in downward_task_list) do
 5:   $n_i$ = downward_task_list.out();
 6:   Compute $D(n_i)$ by (14);
 7:   Get the assigned processor $u_{pr(i)}$ of $n_i$, which has
      been obtained based on HEFT algorithm;
 8:   var $f_{pr(i),hz(i)} = NULL$, $AFT(n_i) = NULL$,
      $E(n_i) = \infty$;
 9:   for (each frequency $f_{k,h}$ in the scope of [$f_{k,\text{low}}$,
      $f_{k,\max}$]) do
10:     Compute EFT$(n_i, u_{pr(i)}, f_{pri(i),h})$ value using
        (15) based on the insertion-based scheduling
        policy;
11:     if (EFT$(n_i, u_{pr(i)}, f_{pr(i),h}) \leqslant D(n_i)$) then
12:       Calculate $E(n_i, pr(i), f_{pr(i),h})$ by (9);
13:       if ($E(n_i, u_{pr(i)}, f_{pr(i),h}) < E(n_i)$) then
14:         $f_{pr(i),hz(i)} = f_{pr(i),h}$;
15:         AFT$(n_i)$ = EFT$(n_i, u_{pr(i)}, f_{pr(i),h})$;
16:         $E(n_i) = E(n_i, u_{pr(i)}, f_{pr(i),h})$;
17:       end if
18:     end if
19:   end for
20: end while
21: Calculate $SL(G)$ by (10);
22: Calculate $E(G)$ by (11).
```

1) In Line 2, all the assigned processors of the tasks have been obtained by using the HEFT algorithm.

2) In Line 6, the deadline $D(n_i)$ of the task n_i has been obtained before assigning it.

3) In Lines 9–19, all the frequencies of processor $u_{pr(i)}$ (the precision is known) are traversed, and the frequency with the minimum energy consumption satisfying the condition EFT$(n_i, u_k, f_{k,h}) \leqslant D(n_i)$ is selected.

4) Given that task n_i scales down the processing frequency within its $D(n_i)$ on the fixed processor, the execution increment of n_i, which is less than the deadline slack $DS(n_i)$, could also be extended to all of its immediate or mediate successors with the same size. Therefore, the final AFT of the exit task would be less than or equal to its deadline. That is, the $SL(G)$ computed by the DECM algorithm is safe and always less than or equal to $D(G)$.

TABLE III
POWER AND FREQUENCY PARAMETERS OF THE PROCESSORS (u_1, u_2, AND u_3)

Processor	$P_{k,\text{ind}}$	$C_{k,\text{ef}}$	m_k	$f_{k,\text{low}}$	$f_{k,\text{max}}$
u_1	0.03	0.8	2.9	0.22	1.0
u_2	0.04	0.8	2.5	0.21	1.0
u_3	0.07	1.0	2.5	0.29	1.0

TABLE IV
DECM-GENERATED TASK ASSIGNMENT OF THE APPLICATION IN FIG. 1

Task	Assigned processor	Frequency	AST	AFT	Deadline	Energy consumption
n_1	u_3	**0.65**	**0**	**13.8461**	**14.0**	**5.6856**
n_3	u_3	**0.79**	**13.8461**	**37.8967**	**38.0**	**15.0247**
n_4	u_2	**0.61**	**22.8461**	**35.9609**	**36.0**	**3.5737**
n_2	u_1	**0.72**	**31.8461**	**49.9017**	**50.0**	**6.1130**
n_5	u_3	**0.99**	**37.8967**	**47.9977**	**48.0**	**10.5574**
n_6	u_2	1.0	35.9609	51.9609	52.0	13.4400
n_9	u_2	**0.71**	**65.9017**	**82.8031**	**83.0**	**6.4193**
n_7	u_3	**0.69**	**47.9977**	**63.9398**	**64.0**	**7.4206**
n_8	u_1	**0.5**	**66.9609**	**76.9609**	**77.0**	**1.3717**
n_{10}	u_2	**0.59**	**87.9609**	**99.8253**	**100.0**	**3.0124**
$SL(G) = 99.8253 < D(G) = 100$, $E(G) = 72.6187$						

HEFT has been proved to perform very competitively, and it has a low time complexity of $O(|N|^2 \times |U|)$. Similarly, DECM also has a low time complexity of $O(|N|^2 \times |U| + |N|^2 \times |F|)$, when the discrete frequencies are introduced. $|F|$ represents the maximum number of discrete frequencies from the lowest to the maximum actual effective frequencies. Therefore, DECM implements the energy consumption minimization without increasing time complexity.

D. Example of the DECM Algorithm

This section illustrates an example to show the results of the DECM algorithm. The frequency-independent dynamic power $P_{k,\text{ind}}$, effective switching capacitance $C_{k,\text{ef}}$, and the dynamic power exponent m_k for all the processors are assumed to be known, and their values are in Table III. The lowest energy-efficient frequency $f_{k,\text{low}}$ for each processor is obtained according to (3). Meanwhile, the maximum frequency $f_{k,\text{max}}$ for each processor is assumed to be 1.0.

Example 1: In this example, $D(G) = 100$ and the values of the power and frequency parameters of the processors are shown in Table III. Table IV lists the task assignment of the parallel application in Fig. 1. Each row represents a task assignment and its corresponding values. For example, n_1 is assigned to u_3 with a frequency of 0.65; the AST of n_1 is 0, and the AFT of n_1 is 13.8461, which is lower than its deadline 14; as a result, the consumed energy for n_1 is 5.6856. For the rows in bold fonts of values, the frequencies of the tasks are scaled down to values lower than the maximum frequency 1.0, and the corresponding energy consumption values are reduced by DECM compared with energy consumption values on the same processors by the HEFT algorithm. Specifically, the AFT of each task does not exceed its deadline by DECM.

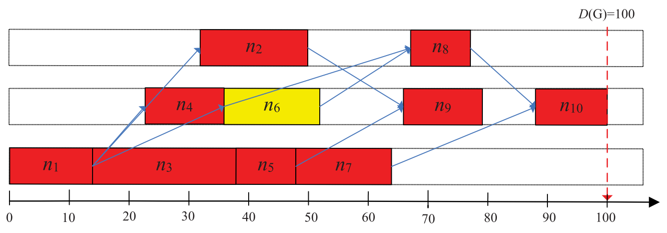


Fig. 3. DECM-generated task scheduling of the application in Fig. 1.

Fig. 3 also shows the scheduling of application G. In this figure, the energy consumption values of the tasks in red boxes are reduced by scaling down their frequencies to lower values, such that their execution time is longer. For example, when HEFT is used, n_9 is assigned to u_2 with a frequency of 1, and $\text{AST}(n_9) = 56$ and $\text{AFT}(n_9) = 68$, as shown in Fig. 2. When DECM is used, n_9 is still assigned to u_2; however, the assigned frequency is scaled down to 0.71, and AST and AFT are changed to $\text{AST}(n_9) = 65.9017$ and $\text{AFT}(n_9) = 82.8031$, respectively, as shown in Fig. 3. This figure also shows that the precedence constraints are not violated among all the tasks and the deadline of the application is also satisfied. Finally, the tasks satisfy the condition $\text{AFT}(n_i) \leqslant D(n_i)$; the DECM-obtained schedule length of the application is lower than its deadline (i.e., $SL(G) = 99.8253 < D(G) = 100$). The final energy consumptions of the application using DECM is $E(G) = 72.6187$, which is also lower than $E_{\text{max}}(G) = 159.49$ when HEFT is used.

E. Latest Finish Time

The DECM-obtained schedule length is safe, but DECM merely considers the energy consumption minimization from a "downward" perspective. If $LB(G)$ is equal to $D(G)$, then DECM is not applicable because $DS(G) = 0$. However, slacks may exist between two adjacent tasks on the same processor, and these slacks can be eliminated or reduced. In this section, the DUECM algorithm is presented to solve the problem from both "downward" and "upward" perspectives. "Upward" means that energy consumption minimization is implemented from the exit task to the entry task according to the descending order of AFT.

The main idea of DUECM is that the DECM-obtained $\text{AFT}(n_i)$ can be extended to $\text{LFT}(n_i)$ because slacks exist between adjacent tasks in the same processor [20], [21]. $\text{LFT}(n_i)$ is calculated by [20], [21]

$$\begin{cases} \text{LFT}(n_{\text{exit}}) = D(G) \\ \text{LFT}(n_i) = \min\left\{ \min\limits_{n_j \in \text{succ}(n_i)} \{\text{AST}(n_j) - c'_{i,j}\}, \text{AST}(n_{dn(i)}) \right\} \end{cases} \tag{17}$$

where $n_{dn(i)}$ represents the downward neighbor task of n_i. For example, $n_{dn(2)} = n_8$ and $n_{dn(9)} = n_{10}$, as shown in Fig. 2. Given that slacks exist in processors, $\text{LFT}(n_i) \geqslant \text{AFT}(n_i)$ for any task n_i. For example, $\text{AFT}(n_9) = 77.89$, as shown in Fig. 3, but $\text{LFT}(n_9)$ should be 83. Therefore, the actual ex-

Algorithm 2: The DUECM Algorithm.

```
Input: $U = \{u_1, u_2, \ldots, u_{|U|}\}$, $G$
Output: $SL(G)$, $E(G)$
1:  Call the DECM algorithm (Algorithm 1) to implement
    downward energy consumption minimization;
2:  Sort the tasks in the list upward_task_list according to
    the descending order of AFT($n_i$) values;
3:  while (tasks exist in upward_task_list) do
4:      $n_i$ = upward_task_list.out();
5:      Calculate LFT($n_i$) by (17);
6:      Calculate new frequency $f_{pr(i),nhz(i)}$ by (18)
        and (19);
7:      Update AET($n_i$) by (20);
8:      Update AFT($n_i$) ← LFT($n_i$);
9:      Update AST($n_i$) ← (LFT($n_i$) − AET($n_i$)) by (21);
10:     Update $f_{pr(i),hz(i)} \leftarrow f_{pr(i),nhz(i)}$;
11:     Calculate $E(n_i)$ by (9);
12: end while
13: Calculate $E(G)$ by (11).
```

ecution time (AET) is changed from $\mathrm{AFT}(n_i) - \mathrm{AST}(n_i)$ to $\mathrm{LFT}(n_i) - \mathrm{AST}(n_i)$, and the new frequency can be obtained as follows:

$$f_{pr(i),nhz(i)} = f_{pr(i),hz(i)} \times \frac{\mathrm{AFT}(n_i) - \mathrm{AST}(n_i)}{\mathrm{LFT}(n_i) - \mathrm{AST}(n_i)}. \tag{18}$$

Given that the lowest frequency to execute a task is $f_{pr(i),\mathrm{low}}$, $f_{pr(i),nhz(i)}$ should be

$$f_{pr(i),nhz(i)} = \max\{f_{pr(i),nhz(i)}, f_{pr(i),\mathrm{low}}\}. \tag{19}$$

Finally, the new $\mathrm{AET}(n_i)$ with the new frequency $f_{pr(i),nhz(i)}$ should be updated as

$$\mathrm{AET}(n_i) = f_{pr(i),hz(i)} \times \frac{\mathrm{LFT}(n_i) - \mathrm{AST}(n_i)}{f_{pr(i),nhz(i)}} \tag{20}$$

and the new AST should be updated as

$$\mathrm{AST}(n_i) = \mathrm{AFT}(n_i) - \mathrm{AET}(n_i). \tag{21}$$

F. DUECM Algorithm

On the basis of above equations, the DUECM algorithm described in Algorithm 2 is proposed. The core details of DUECM are explained as follows.

1) In Line 2, the tasks in the list upward_task_list is sorted according to the descending order of $\mathrm{AFT}(n_i)$ values, which are obtained by DECM (i.e., from exit to entry tasks).

2) In Lines 4–7, $\mathrm{LFT}(n_i)$; consequently, the frequency and $\mathrm{AET}(n_i)$ are updated.

3) In Lines 8–11, $\mathrm{AFT}(n_i)$ and $\mathrm{AST}(n_i)$ are updated.

4) In Line 13, the new $E(G)$ is calculated.

DUECM has low time complexity of $O(|N|^2 \times |U| + |N|^2 \times |F|)$. In fact, the time complexity for Lines 2–13 is merely $O(|N|^2)$. Corresponding to DECM, Line 1 accounts for the major time complexity.

TABLE V
DUECM-Generated Task Assignment of the Application in Fig. 1

Task	Processor	Frequency	AST	AFT	LFT	Energy
n_{10}	u_2	**0.58**	**87.9609**	**100.0**	**100.0**	**2.9490**
n_9	u_2	**0.54**	**65.9017**	**87.9609**	**87.9609**	**4.6638**
n_8	u_1	**0.5**	**66.9609**	**76.9609**	**76.9609**	**1.3717**
n_7	u_3	**0.48**	**48.0442**	**70.9609**	**70.9609**	**5.2622**
n_6	u_2	1.0	35.9609	51.9609	51.9609	13.4400
n_2	u_1	0.72	31.8461	49.9017	49.9017	6.1130
n_5	u_3	0.99	37.9432	48.0442	48.0442	10.5574
n_3	u_3	0.79	13.8925	37.9432	37.9432	15.0247
n_4	u_2	0.61	22.8461	35.9609	35.9609	3.5737
n_1	u_3	0.65	0	13.8461	13.8461	5.6856
$SL(G) = D(G) = 100$, $E(G) = 68.6415$						

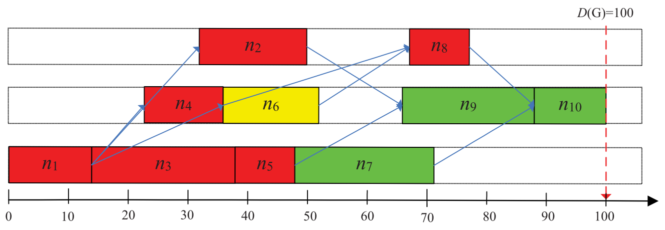


Fig. 4. DUECM-generated scheduling of the application in Fig. 1.

G. Example of the DUECM Algorithm

Example 2: In this example, $D(G) = 100$ and values of the power and frequency parameters of the processors are the same values shown in Table III. Table V presents the task assignment of the parallel application in Fig. 1. For the rows in bold fonts of values, the tasks's ASTs or AFTs of the tasks are changed using the DUECM algorithm (see Table V).

Fig. 4 also shows the DUECM-derived scheduling of application G. For example, when DECM is used, n_9 is assigned to u_2 with a frequency of 0.71, and $\mathrm{AST}(n_3) = 65.9017$ and $\mathrm{AFT}(n_3) = 82.8031$ as shown in Fig. 3. When DUECM is used, n_9 is still assigned to u_2; however, the assigned frequency is scaled down to 0.54, and AST and AFT are changed to $\mathrm{AST}(n_3) = 65.9017$ and $\mathrm{AFT}(n_3) = 87.9609$, respectively, as shown in Fig. 4 using DUECM. This figure also shows that the precedence constraints are not violated among all the tasks, and the deadline of the application is also satisfied.

All the tasks can satisfy the condition $\mathrm{AFT}(n_i) \leqslant \mathrm{LFT}(n_i)$, and the DUECM-generated schedule length of the application is equal to its deadline (i.e., $SL(G) = 100 = D(G) = 100$). The final energy consumption of the application using the DUECM is $E(G) = 68.6415$, which is lower than the $E(G)$ value (72.6187) obtained using the DECM algorithm. Finally, as roughly shown in Fig. 4, DECM is to minimize energy consumption values of the tasks near the entry task, and DUECM is to minimize energy consumption values of the tasks near the exit task.

H. Implementation Issues

Currently, the mainstream manufacturers, such as Intel, ARM, and AMD, provide processors that support for DVFS processor technology, such as enhanced Intel SpeedStep technology for Intel, intelligent energy manager, and adaptive voltage scaling for ARM, and PowerNow for AMD. A typical DVFS-enabled heterogeneous system can be implemented as follows.

1) Select concrete processors (e.g., AMD Athlon-64, Intel Pentium M, AMD Opteron 2218, and AMD Turion MT-34) and build a distributed system, where these processors are mounted on the same communication bus, such as controller area network. Note that different processors have different DVFS levels.

2) Set the workload and frequency on each time range of each processor for the parallel application according to the schedule results using the proposed algorithm. Note that the processor chip clock varies with frequency.

3) Calculate corresponding voltage based on the new frequency, and then inform the power management module to adjust the CPU voltage. Note that special power management chips, such as Freescale's MC13783 or NS's PowerWise-enabled family of power management chips, are required because they are capable of supporting very small voltage regulation (25 mV) and enabling voltage regulation in a very short time (tens of microseconds).

4) In addition, when adjusting the frequency and voltage, pay special attention to the order of adjustment. When decreasing the frequency from high to low, one should first drop the frequency, and then drop the voltage; on the contrary, when raising the frequency, one should first raise the voltage, and then raise the frequency.

Some problems should be concerned in the real DVFS-enabled systems.

1) There are communication, network card, and voltage/frequency switching overheads for the application, and these overhead would affect the precision of schedule.
2) Tasks state saving and recovery, and phase-locked loop's shock and stability would affect the precision of DVFS.
3) In addition to the energy consumption of the processors, other components also generate considerable energy consumption, such as memory, interconnect networks, power supplies, etc. These components would lead to increased energy consumption of the system. Moreover, leakage current is increasing.

V. Experiments

A. Experimental Metrics and Data

Considering that this study aims to minimize the energy consumption of real-time parallel application, we use the actual energy consumption $E(G)$ [see (11)] and the final schedule length $SL(G)$ of the application as performance metrics to get very intuitive comparison results. The algorithms compared with the proposed DUECM are the state-of-the-art EES [20] and DEWTS [21]. The values of the processor and application parameters taken from [11] and [12] are as follows: 10 ms $\leqslant w_{i,k} \leqslant$ 100 ms, 10 ms $\leqslant c_{i,j} \leqslant$ 100 ms, 0.03 $\leqslant P_{k,\text{ind}} \leqslant$ 0.07, 0.8 $\leqslant C_{k,\text{ef}} \leqslant$ 1.2, 2.5 $\leqslant m_k \leqslant$ 3.0, and $f_{k,\max} = 1$ GHz. All frequencies are discrete, and the precision is 0.1 GHz. These parameter values basically reflect the characters of some high-performance embedded processors, such as Intel Mobile Pentium III and ARM Cortex-A9.

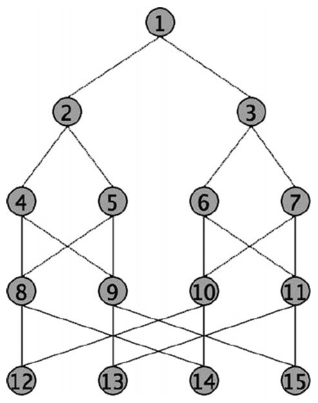


Fig. 5. Example of the fast Fourier transform application with $\rho = 4$.

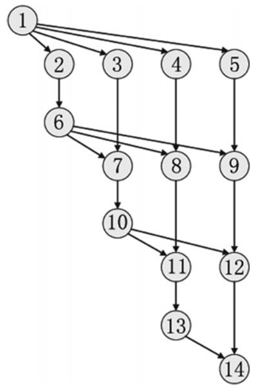


Fig. 6. Example of the Gaussian elimination parallel application with $\rho = 5$.

Two types of real parallel applications (fast Fourier transform and Gaussian elimination) are used to provide experimental results. Fig. 5 shows an example of the fast Fourier transform application with $\rho = 4$. ρ is used as the size of the fast Fourier transform application, and the total number of tasks is $|N| = (2 \times \rho - 1) + \rho \times \log_2{}^{\rho}$, where $\rho = 2^y$ for some integer y [16]. In the fast Fourier transform application with the size ρ, ρ exit tasks exist. To adapt the application model to this study, just a virtual exit task is added, and the last ρ tasks are set as the immediate predecessor tasks of the virtual task. Fig. 6 shows an example of the Gaussian elimination application with $\rho = 5$, and the total number of tasks is $|N| = \frac{\rho^2+\rho-2}{2}$ [16].

Similar to most studies [11], [12], [20], [21], the parallel applications are executed on a simulated heterogeneous system based on a real processor and application parameter values. A main advantage of simulation is that it can greatly reduce development cost during the design phase and effectively provide certain optimization guide to the implementation phase.

B. Fast Fourier Transform Application for Varying Deadlines

Experiment 1: This experiment is conducted to compare the actual schedule length and final energy consumption of the fast

TABLE VI
FINAL ENERGY CONSUMPTION (UNIT: KWS) AND ACTUAL SCHEDULE LENGTH (UNIT: MS) OF THE FAST FOURIER TRANSFORM APPLICATION WITH $\rho = 256$ (I.E., $|N| = 2560$) FOR VARYING DEADLINE CONSTRAINTS

HEFT [16]				EES [20]		DEWTS [21]		DUECM	
$\|N\|$	$E_{\max}(G)$	$LB(G)$	$D(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
2560	46874.01	1192	1192	37224.61	1192	37038.36	1192	37224.61	1192
2560	46874.01	1192	1311.2	35141.05	1311.2	35965.29	1311.2	28130.93	1311.2
2560	46874.01	1192	1430.4	32330.34	1430.4	38126.84	1430.4	24193.81	1430.4
2560	46874.01	1192	1549.6	30178.00	1549.6	39440.82	1549.6	22051.42	1549.6
2560	46874.01	1192	1668.8	27663.53	1668.8	40283.02	1668.8	20365.40	1668.8

TABLE VII
FINAL ENERGY CONSUMPTIONS (UNIT: KWS) AND ACTUAL SCHEDULE LENGTHS (UNIT: MS) OF FAST FOURIER TRANSFORM APPLICATIONS WITH THE DEADLINE CONSTRAINT $D(G) = LB(G) \times 1.4$ FOR VARYING NUMBERS OF TASKS

HEFT [16]				EES [20]		DEWTS [21]		DUECM	
$\|N\|$	$E_{\max}(G)$	$LB(G)$	$D(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
224	4954.46	711	995.4	3016.37	995.4	5004.42	995.4	2178.49	995.4
512	11362.46	880	1232	6872.98	1232	10327.18	1232	4863.11	1232
1152	22985.53	1051	1471.4	13344.87	1471.4	20510.57	1471.4	9586.49	1471.4
2560	47255.23	1174	1643.6	26506.49	1643.6	42132.36	1643.6	19445.66	1643.6

Fourier transform application for varying deadline constraints. The size of the application is limited to $\rho = 256$ (i.e., $|N| = 2560$) and change $D(G)$ from $LB(G) \times 1.0$ to $LB(G) \times 1.4$.

Table VI shows the results of the fast Fourier transform application for varying deadline constraints by using the three algorithms. The schedule lengths obtained by the three algorithms are equal to the given deadlines, that is, all the algorithms can satisfy the deadline constraints by using LFT and the "upward" approach. The maximum energy consumption for HEFT is $E_{\max}(G) = 45667.23$ kWs (1 J = 1 W $\times$ 1 s = 1 kWs), and all the compared three algorithms can reduce the energy consumption to a certain extent. When $D(G) = LB(G) \times 1.0$, DEWTS allows for a slightly lower energy consumption than EES and DUECM. The reason for the above results can be explained as follows: 1) when using EES and DUECM, the LFT of the exit task cannot be extended in this case (i.e., $D(G) = LB(G) \times 1.0$), and only a small part of tasks can be optimized and the total energy consumption reduction is limited; 2) when using DEWTS, as the deadline is equal to lower bound, only a few processors are turned OFF, then all the tasks in fact experience a reassignment on processors; such process is equivalent to using a low energy consumption task rescheduling algorithm. That is, DEWTS implements a low energy consumption task assignment, such that it can reduce more energy consumption than EES and DUECM in this case. Although this study can also include such a feature into DUECM, it is practically unrealistic and the algorithm requires high time complexity. Moreover, DEWTS consumes more energy consumption than DUECM in other cases. Particulary, the energy consumption using DUECM is about half of that using DEWTS when $D(G) = LB(G) \times 1.4$. Note that DUECM can generally save more energy consumption than EES and DEWTS except for the case that $D(G) = LB(G) \times 1.0$. Specifically, as the deadlines increase, DUECM-generated energy consumption values are reduced gradually from 37224.61 to 20365.40 kWs, whereas DEWTS (from 37038.36 to 40283.02 kWs) not only fail to further reduce energy consumptions but also increase energy consumptions in some cases. When $D(G) = LB(G) \times 1.4$, DUECM is better than EES and DEWTS by 26.38% and 49.44%, respectively, in terms of energy consumption minimization. Such results indicate that: 1) the energy consumption reduction space is limited by merely applying the "upward" approach even if the processors are turned OFF and the tasks moved; 2) synthetically applying both "downward" and "upward" approaches can reduce energy consumption more than by merely applying the "upward" approach, and the superiority of the former is more significant when the deadline-slack is large.

C. Fast Fourier Transform Application for Varying Scales

Experiment 2: To observe the performance on different scales of applications, an experiment is conducted to compare the final energy consumption values and actual schedule lengths of fast Fourier transform applications for varying numbers of tasks. Let $D(G) = LB(G) \times 1.4$ and ρ is changed from 32 to 256, that is, the numbers of tasks vary from 224 (small scale) to 2560 (large scale).

Table VII shows the results of the fast Fourier transform applications for varying numbers of tasks by using the three algorithms. Similar to the results of *Experiment 1*, the schedule lengths obtained using the algorithms are equal to the given deadlines, and the energy consumption values are reduced to a certain extent. For all the three algorithms, the generated energy consumption increases gradually with an increase in the number of tasks. In all cases, DUECM can reduce consumption more than EES and DEWTS and outperform the two state-of-the-art algorithms by 27.78% and 56.47% in the best case ($|N| = 224$). These results further verify that DUECM, which is synthetically implements the "downward" and "upward" approaches, can more efficiently minimize energy consumption than EES and DEWTS, both of which apply the "upward" approach.

D. Gaussian Elimination Application for Varying Deadlines

Experiment 3: This experiment is conducted to compare the actual schedule lengths and final energy consumption values of a Gaussian elimination application for varying deadline constraints. The size of the application is limited to $\rho = 71$ (i.e., $|N| = 2555$, which is approximately equal to the number of tasks of the fast Fourier transform application in *Experiment 1*), and change $D(G)$ from $LB(G) \times 1.0$ to $LB(G) \times 1.4$.

Table VIII shows the results of the Gaussian elimination application for varying deadline constraints by using the three algorithms. Compared with the results of *Experiment 1* in Table VI, the Gaussian elimination application has a longer

TABLE VIII
FINAL ENERGY CONSUMPTIONS (UNIT: KWS) AND ACTUAL SCHEDULE LENGTHS (UNIT: MS) OF THE GAUSSIAN ELIMINATION APPLICATION WITH $\rho = 71$ (I.E., $|N| = 2555$) FOR VARYING DEADLINE CONSTRAINTS

HEFT [16]				EES [20]		DEWTS [21]		DUECM	
$\|N\|$	$E_{\max}(G)$	$LB(G)$	$D(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$	$E(G)$	$SL(G)$
2555	50569.59	7212	7212	33656.17	7212	33181.77	7212	33656.17	7212
2555	50569.59	7212	7933.2	32280.59	7933.2	60767.70	7933.2	29728.24	7933.2
2555	50569.59	7212	8654.4	30259.70	8654.4	58588.55	8654.4	26700.64	8654.4
2555	50569.59	7212	9375.6	27587.61	9375.6	56166.23	9375.6	25101.30	9375.6
2555	50569.59	7212	10096.8	24574.10	10096.8	57862.86	10096.8	22505.48	10096.8

schedule length than the fast Fourier transform application in all cases. For example, the lower bounds of the fast Fourier transform and Gaussian elimination applications with the use of HEFT are 1192 and 7212 ms, respectively. The lower bound of the former is merely 16.5% of the latter.

Similar to the results of *Experiment 1* in Table VI, DUECM can reduce energy consumption more than EES and DEWTS, except for the case in which $D(G) = LB(G) \times 1.0$. When $D(G) = LB(G) \times 1.4$, DUECM is better than EES and DEWTS by 8.42% and 61.1%, respectively, in terms of energy consumption minimization. Although the results of the Gaussian elimination application are lower than those of the fast Fourier transform application on the same scale, their overall trends are similar. First, with an increase in the deadline, the generated energy consumption using DUECM is reduced gradually, whereas DEWTS not only fails to reduce energy consumption but also increases energy consumption in some cases. The results show that DUECM is effective in different types of parallel applications, and its effectiveness is stable.

All the results of the fast Fourier transform and Gaussian elimination applications show that the proposed DUECM illustrates more effective energy consumption minimization than EES and DEWTS in a variety of different conditions with different deadline constraints.

VI. CONCLUSION

An effective energy consumption minimization algorithm called DUECM is presented for real-time parallel applications. First, DUECM is designed for heterogeneous systems, and it has low time complexity. Second, DUECM demonstrates more effective energy consumption minimization than the state-of-the-art algorithms as it reduces energy consumption by applying both "downward" and "upward" approaches. Third, DUECM is highly efficient for different types of real parallel applications under a variety of different conditions. DUECM can effectively facilitate the energy-aware design for real-time parallel applications in heterogeneous environments. In future work, a "global" approach by reclaiming slack time on different processors for each task would be considered. In addition, heterogeneous DVFS-enabled systems are not very popular in some embedded systems, so it is also necessary to study low energy consumption task scheduling method for non-DVFS environments. The real systems will be implemented to run real applications and evaluate their effectiveness in practice.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers whose constructive comments have helped to improve the manuscript.

REFERENCES

[1] A. Schranzhofer, J. J. Chen, and L. Thiele, "Dynamic power-aware mapping of applications onto heterogeneous MPSoC platforms," *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 692–707, Nov. 2010.

[2] J. Castrillon, R. Leupers, and G. Ascheid, "Maps: Mapping concurrent dataflow applications to heterogeneous MPSoCs," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 527–545, Feb. 2013.

[3] M. Hu, J. Luo, Y. Wang, M. Lukasiewycz, and Z. Zeng, "Holistic scheduling of real-time applications in time-triggered in-vehicle networks," *IEEE Trans. Ind. Informat.*, vol. 10, no. 3, pp. 1817–1828, Aug. 2014.

[4] I. Ukhov, P. Eles, and Z. Peng, "Temperature-centric reliability analysis and optimization of electronic systems under process variation," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 11, pp. 2417–2430, Nov. 2015.

[5] M. Bao, A. Andrei, P. Eles, and Z. Peng, "Temperature-aware idle time distribution for leakage energy optimization," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 7, pp. 1187–1200, Jul. 2012.

[6] A. Andrei, P. Eles, Z. Peng, and M. T. Schmitz, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 15, no. 3, pp. 262–275, Mar. 2007.

[7] A. Andrei, P. Eles, O. Jovanovic, M. Schmitz, J. Ogniewski, and Z. Peng, "Quasi-static voltage scaling for energy minimization with time constraints," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 1, pp. 10–23, Jan. 2011.

[8] L. Chandnani and H. K. Kapoor, "Formal approach for DVS-based power management for multiple server system in presence of server failure and repair," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 502–513, Feb. 2013.

[9] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *IEEE Trans. Comput.*, vol. 61, no. 12, pp. 1668–1681, Dec. 2012.

[10] B. Zhang, R. Simon, and H. Aydin, "Harvesting-aware energy management for time-critical wireless sensor networks with joint voltage and modulation scaling," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 514–526, Feb. 2013.

[11] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 316–328, Aug. 2010.

[12] B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 2, pp. 99–109, Mar. 2013.

[13] G. Xie, G. Zeng, L. Liu, R. Li, and K. Li, "High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems," *J. Syst. Archit.*, vol. 70, pp. 3–14, Oct. 2016.

[14] D. P. Bunde, "Power-aware scheduling for makespan and flow," *J. Scheduling*, vol. 12, no. 5, pp. 489–500, Oct. 2009.

[15] C. Rusu, R. Melhem, and D. Mossé, "Maximizing rewards for real-time applications with energy constraints," *ACM Trans. Embedded Comput. Syst.*, vol. 2, no. 4, pp. 537–559, Nov. 2003.

[16] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Aug. 2002.

[17] M. A. Khan, "Scheduling for heterogeneous systems using constrained critical paths," *Parallel Comput.*, vol. 38, no. 4, pp. 175–193, 2012.

[18] G. Xie, R. Li, and K. Li, "Heterogeneity-driven end-to-end synchronized scheduling for precedence constrained tasks and messages on networked embedded systems," *J. Parallel Distrib. Comput.*, vol. 83, pp. 1–12, Sep. 2015.

[19] K. Li, "Power and performance management for parallel computations in clouds and data centers," *J. Comput. Syst. Sci.*, vol. 82, no. 2, pp. 174–190, Mar. 2016.

[20] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2012, pp. 781–786.

[21] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *J. Grid Comput.*, vol. 14, no. 1, pp. 55–74, Mar. 2016.

[22] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Mobile Computing*. New York, NY, USA: Springer-Verlag, 1996, pp. 449–471.

[23] J. Zhuo and C. Chakrabarti, "Energy-efficient dynamic task scheduling algorithms for DVS systems," *ACM Trans. Embedded Comput. Syst.*, vol. 7, no. 2, pp. 421–434, Feb. 2008.

[24] J.-J. Han, X. Wu, D. Zhu, H. Jin, L. T. Yang, and J.-L. Gaudiot, "Synchronization-aware energy management for VFI-based multicore real-time systems," *IEEE Trans. Comput.*, vol. 61, no. 12, pp. 1682–1696, Jun. 2012.

[25] J. Barnett, "Dynamic task-level voltage scheduling optimizations," *IEEE Trans. Comput.*, vol. 54, no. 5, pp. 508–520, May 2005.

[26] K. Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1484–1497, Jul. 2008.

[27] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2014.

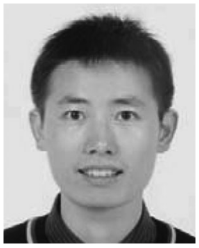
[28] Z. Zong, A. Manzanares, X. Ruan, and X. Qin, "EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters," *IEEE Trans. Comput.*, vol. 60, no. 3, pp. 360–374, Jan. 2011.

[29] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1374–1381, Jun. 2011.

Guoqi Xie (M'15) received the Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2014.

He was a Postdoctoral Researcher at Nagoya University, Japan, from 2014 to 2015. Since 2015, he has been a Postdoctoral Researcher at Hunan University. His major interests include embedded and real-time systems, parallel and distributed systems, software engineering, and methodology.

Dr. Xie received the Best Paper Award from ISPA 2016. He is a member of ACM and CCF.



Junqiang Jiang is currently working toward the Ph.D. degree in computer science and engineering at Hunan University, Changsha, China.

His research interests include energy-efficient systems and parallel and distributed systems.

Yan Liu received the Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2010.

He is an Assistant Professor in the College of Computer Science and Electronic Engineering, Hunan University. His major interests include computer architectures and embedded computing systems. He is a member of CCF.

Renfa Li (M'05–SM'10) received the Ph.D. degree in electronic engineering from Huazhong University of Science and Technology, Wuhan, China, in 2003.

He is a Professor of computer science and electronic engineering, and the Dean of the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, China. He is also an expert committee member of the National Supercomputing Center, Changsha. His major interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of things. He is a member of the council of CCF and a senior member of ACM.

Keqin Li (M'90–SM'96–F'15) received the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things, and cyber-physical systems. He has published more than 460 journal articles, book chapters, and refereed conference papers.

Dr. Li has received several best paper awards. He is currently or has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.