Journal of Systems Architecture xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

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

# On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip

Ya-Shu Chen *, Ming-Yang Chen

*Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei City, Taiwan, ROC*

ARTICLE INFO

*Article history:*
Received 15 February 2012
Received in revised form 14 May 2012
Accepted 28 May 2012
Available online xxxx

*Keywords:*
Energy-efficient
Dual-core
Real-time
Task scheduling
Embedded system

ABSTRACT

On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip is a challenging problem due to precedence constraints and the varied properties of the general-purpose processor core and the synergistic processor core. This study proposes an on-line heterogeneous dual-core energy-efficient scheduling framework for dynamic workloads with real-time constraints. The energy efficiency ratio is presented to manage energy consumption while considering of the varied properties of the cores, while precedence constraints among the tasks are dealt with through interaction between bandwidth servers. This framework is configurable for low energy consumption and high system utilization. The capability of the proposed methodology is evaluated by a series of experiments and the results obtained are encouraging.



## 1. Introduction

Energy consumption minimization is a critical problem for modern multimedia hand-held systems. Thus, to improve performance and power efficiency, such systems can be equipped with a heterogeneous dual-core system-on-a-chip with support for dynamic voltage scaling; for example, one general purpose processor core and one synergistic processor core (as a coprocessor) [1–3]. In contrast to homogeneous multi-core systems, in a heterogeneous dual-core system, the general purpose processor core and the synergistic processor core have different properties, such as fast execution time, significant context overhead and a larger peak power in the synergistic processor core. To meet the quality of service requirement, applications are implemented as several functions to be executed on a specific core on such system. Nevertheless, the energy-efficient real-time task scheduling problem in a heterogeneous dual-core system-on-a-chip is complicated by the precedence constraints of functions and the trade-off between energy consumption and response time. In this paper, we attempt to minimize energy consumption, and meet the timing constraints of applications with a fast admission for on-line dynamic workload.

In the past decade, a lot of research has been performed for energy-aware real-time scheduling in a homogenous multi-core environment with independent task sets, e.g., [4–9]. Most researchers propose load balance methods for each core based on partition dispatchers with energy considerations [4–6,9], and some are based on a global strategy, e.g., [7,8]. However, few researchers have focused on using a task set with a precedence constraint; the research work closest to ours is [10,11], which manage the streaming application with energy consideration on homogeneous multi-core systems.

Although the energy-efficient real-time scheduling problem has been well studied, there are relatively few results that provide energy-efficient real-time task scheduling in heterogeneous multi-core systems. The related work most similar to ours, [12], presented integer linear programming to resolve real-time scheduling in the form of optimization problems. Without considering energy efficiency, [13,14] proposed a scheduling framework to deal with task precedence constraints between cores. Some works, namely [15–18] extended the resource management algorithm to deal with the precedence constraint between a general purpose processor core and a non-preemptive synergistic processor core, such as a digital signal processor (DSP)/graphics processing unit (GPU). Some works [19,20] proposed the scheduling algorithm to minimize the communication overhead for streaming applications on multi-core systems.

To make the resource management algorithm energy-efficient, some previous research works have also proposed energy-efficient real-time task synchronization on single core, e.g., [21–25]. The dual speed (DS) algorithm [21] presents two alternative speeds for task execution: one without blocking, and the other with blocking. Some works extend the idea of dual speed algorithms to

* Corresponding author.
*E-mail addresses:* yschen@mail.ntust.edu.tw, M9807429@mail.ntust.edu.tw (Y.-S. Chen).

1383-7621/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2012.05.004

propose the multi-speed concept, which assigns different speeds for each blocked task [22,23]. With preemptive critical section, [24] presented the frequency inheritance concept, and [25] proposed frequency locking for a switch overhead model. These approaches can be directly applied to provide energy efficiency for a heterogeneous dual-core system-on-a-chip by cooperating with the resource management algorithm. However, the longer blocking time from non-preemptible execution in a synergistic processor core might increase the energy dramatically and decrease the system utilization.

This work is motivated by the need for energy-efficient real-time task scheduling in a heterogeneous dual-core system-on-a-chip, and the difficulty imposed by the trade-off between priority inversion management and energy consumption. The objective is to minimize the energy consumption of a given task set, provided that the schedulability of tasks is guaranteed. The concept of assigning a energy efficiency ratio with bandwidth reservation framework is proposed. Fast frequency assignment and admission control are presented to manage on-line dynamic workloads in heterogeneous dual-core systems. Instead of resolving energy-efficient real-time task scheduling in the form of optimization problems, the proposed framework is configurable for low energy consumption and high system utilization. Two separate frequency scaling techniques for the general purpose processor core and the synergistic processor core are proposed according to their different properties. A series of extensive simulations have also been performed to obtain comparison studies using different workloads, configurations, and scheduling algorithms.

The rest of this paper is organized as follows. Section 2 defines the terminologies and system architecture under consideration. Section 3 proposes the algorithm for on-line energy-efficient real-time task scheduling in heterogeneous dual-core systems, and presents the concepts of the energy efficiency ratio and admission control. Section 4 provides a performance evaluation of the algorithm through extensive simulations. Section 5 concludes this work.

## 2. System model

We are interested in energy-efficient real-time scheduling of periodic tasks in a heterogeneous dual-core system-on-a-chip, where a general purpose processor core is supported by one synergistic processor core, such as a DSP for signal processing or a GPU for graphics processing. For simplicity, we assume that the processor represents the general purpose processor core, and the coprocessor represents the synergistic processor core. The power consumption functions of each core [26] can be modeled as a resultant dynamic power consumption, which arises due to switching activity in a circuit, and a leakage power consumption, which is present even when no logic operations are performed. The power function of the processor **p** is $P_{\mathbf{p}} = P_{\mathbf{p}}(f) + \beta_{\mathbf{p}} = \alpha_{\mathbf{p}} V_{dd}^2 f + \beta_{\mathbf{p}}, f = \kappa \frac{(V_{dd}-V_t)^2}{V_{dd}}$, where $\alpha_{\mathbf{p}}, V_t, V_{dd}$, $\kappa$, and $\beta_{\mathbf{p}}$ denote the effective switch capacitance, the threshold voltage, the supply voltage, the hardware design-specific constant, and the leakage power, respectively. The power function of the coprocessor **cop** can also be modeled by $P_{\mathbf{cop}} = \alpha_{\mathbf{cop}} V_{dd}^2 f + \beta_{\mathbf{cop}}$, with specific power characteristics $\alpha_{\mathbf{cop}}$ and $\beta_{\mathbf{cop}}$ of the coprocessor. In this paper, we assume that the frequency $f$ of each core can be selected independently as any frequency in a given range $[f^{\min}, f^{\max}]$, where $f^{\min}$ is larger than the lowest speed required to eliminate the effect of leakage power consumption [27]. Further extensions of this work for cores with discrete available frequencies can be performed using approaches similar to those reported in [28,29]. The effects of discrete frequencies on the proposed framework are evaluated in the experiment section.

We have further assumed that the processor and the coprocessor communicate using dedicated shared memory and a mailbox. The mailbox is used for sending requests from the processor to the coprocessor, and vice versa. Under the shared memory model, the worst-case communication time can be bounded by or analyzed as a part of the worst-case execution time [30,31]. This study simply considers the communication cost as a part of the execution time without special identification. For more details on the communication architecture implementation, refer to the approaches proposed in [15,16] for DSPs and [17,32] for GPUs. Further examination for minimizing the communication overhead and memory usage for streaming applications can be referred to the mechanism proposed in [19,20].

Most applications are executed in a heterogeneous dual-core system with timing constraints, which are the arrival time, worst-case execution cycle, response time, and relative deadline of a task $\tau_i$ denoted by $a_i, c_i, r_i$, and $D_i$ in this paper. A periodic task $\tau_i$ is a collection of $n_i$ subtasks with a partial order. If a subtask appears before another in the partial order, then the latter cannot start until the former finishes its execution. Each subtask $\tau_{i,j}$ is statically assigned to execute on a processor $p$ or a coprocessor $cop$, and the worst-case execution cycles of $\tau_{i,j}$ is known a priori as $c_{i,j}$. We assume that the execution of subtasks of a task is interleaved between the processor and coprocessor. For example, when the subtask $\tau_{i,j}$ is executed on the processor, then the subtask $\tau_{i,j+1}$ is executed on the coprocessor. The subtasks of a task executed on the processor and coprocessor are referred to as *processor subtasks* and *coprocessor subtasks*. The total worst-case execution cycles of the processor subtasks and the coprocessor subtasks of $\tau_i$ are denoted as $c_i^{\mathbf{p}}$ and $c_i^{\mathbf{cop}}$, respectively.

Task scheduling on the processor is preemptive unless resource conflicts are involved. To avoid significant context-switch overheads on the coprocessor, subtask executions are usually non-preemptible [33,34]. To perform inter-process communication, tasks might share resources with each other. A piece of code is known as a *critical section* if it accesses the shared resource that requires exclusive access. In this study, the critical section is predefined in the corresponding code of each task, and a unique binary semaphore is used to ensure the exclusive entry to the critical section. Before a task accesses any resource, the corresponding semaphore must be locked. When access to the resource or coprocessor has failed due to them being used by a lower priority subtask, the subtask is said *to be blocked*. Depending on the adopted resource manager and the run-time situation, each subtask might suffer from a different amount of *blocking time* from some lower priority subtasks, due to access conflicts.

The system energy is expressed by $E_{\mathbf{system}} = E_{\mathbf{p}} + E_{\mathbf{cop}}$. In this paper, the energy consumption of a task set $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ in the system is defined as the energy consumption of tasks in the *hyperperiod* of **T**, denoted by *HP*, where *HP* is the least common multiple of the task period in **T**. In order to meet the deadline constraint of each task, the low frequencies $f_{\mathbf{p}}^{\mathbf{L}}$ and $f_{\mathbf{cop}}^{\mathbf{L}}$ are assigned to execute the processor and coprocessor subtasks, respectively. To eliminate the blocking effect, the high frequencies $f_{\mathbf{p}}^{\mathbf{H}}$ and $f_{\mathbf{cop}}^{\mathbf{H}}$ are assigned so that no deadline violation is possible when there is a block.

## 3. Energy-efficient heterogeneous dual-core scheduling framework

### 3.1. Framework

In this research, we explore energy-efficient real-time task scheduling for a heterogeneous dual-core system. To resolve

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004

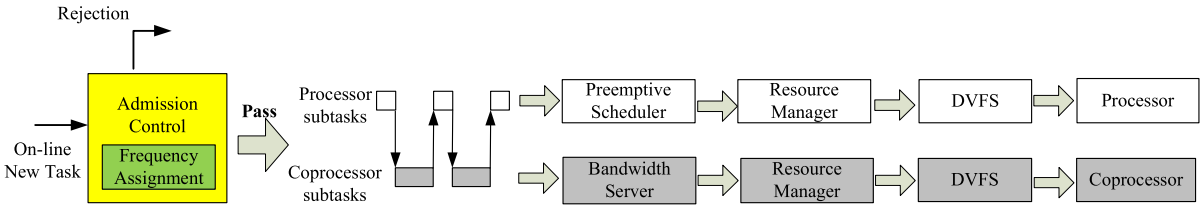



**Fig. 1.** The energy-efficient heterogeneous dual-core scheduling framework.

the precedence constraints of tasks and non-preemptible task execution on the coprocessor, prior work in [13] has proposed a dual-core scheduling (DCS) framework. Under DCS, bandwidth reservation is applied to deal with the precedence constraint between subtasks. To trade-off the context switch overhead and longer non-preemptible executions on a coprocessor, preemption points are inserted into coprocessor subtasks through a compiler so that coprocessor subtasks are scheduled in a semi-preemptive manner.

In consideration of resource sharing and energy-efficiency, we propose the framework shown in Fig. 1, which extends existing work in [13]. During the on-line operation, each new task is tested by admission control. If the task passes the admission control, the proper frequencies for the processor and the coprocessor are assigned to execute tasks; and the frequency assignment should take the energy efficiency ratio and deadline guarantee into account. The arrival time of each subtask is unpredictable and dependent upon the completion time of the previous subtask; and so, each subtask is scheduled on the processor and the coprocessor by a preemptive scheduler with density assignment and bandwidth servers, respectively. After subtasks are assigned priorities by the corresponding scheduler/server, the resource manager allocates resources according to the current priorities of the subtasks. The dynamic voltage frequency scaling operation on the processor and the coprocessor are then invoked to execute scheduled subtasks. In the following sections, we present a detailed protocol which includes scheduling policies, resource allocation rules, and frequency switching conditions. The energy efficiency ratio is then presented to configure the low energy consumption and high system utilization in Section 3.3. The admission control and frequency assignment for this framework will be presented in Section 3.4.

### 3.2. Protocol

The details of energy-efficient heterogeneous dual-core scheduling *EHDS* are shown in Algorithm 1, whereby the deadline assignment and task scheduling are the same as in the scheduling algorithm DCS proposed by Chen [13]. The idea is to schedule each task by a corresponding bandwidth reservation on each core to resolve the precedence constraint between subtasks. By using separate schedulers, each subtask is assigned a local deadline when it has arrived. The admission control can be performed by checking the end-to-end deadline and utilization bounds of the bandwidth reservation. The admission control is listed in Section 3.4. To reduce the blocking time from non-preemptible execution and the context switch overhead on the coprocessor, DCS inserts preemption points using a compiler into the coprocessor subtasks. Subtasks are scheduled in a semi-preemptive manner on the coprocessor. Unlike in DCS, to minimize energy consumption we assign the processor density and server size using the energy efficiency ratio, labeled *EER*. The detailed processor density and server size assignment is presented in Section 3.3.

Algorithm 1 EHDS

I Deadline assignment:
- Each task $\tau_i$ is assigned a processor density $S_i^{\mathbf{p}}$ on the processor and a bandwidth server with size $S_i^{\mathbf{cop}}$ according to the energy efficiency ratio *EER*.
- The deadlines of processor and coprocessor subtasks are assigned by the density and the server size of the corresponding task, respectively.
- The deadline assignment follows the concept of bandwidth server.

II Scheduling rule: let $\Pi(t)$ denote the maximum preemption ceiling of resources currently locked by tasks other than task $\tau_i$ at time $t$.
- Each task is accepted when it has passed admission control. Then each subtask of the task is assigned a local deadline by deadline assignment, and each subtask inherits the preemption level of the corresponding task
- After a subtask is released, it is blocked from starting execution until its preemption level is higher than the current $\Pi(t)$.
- Subtasks are scheduled with their local deadlines on the processor by a preemptively deadline-driven scheduler.
- Subtasks are scheduled by the corresponding bandwidth server at preemption points of the coprocessor.

III Resource allocation: whenever there is a subtask request for a resource, it is allocated the resource.

IV Deadline inheritance: when some subtask is blocked from starting, the blocking subtask inherits the shortest deadline of the blocked subtasks.

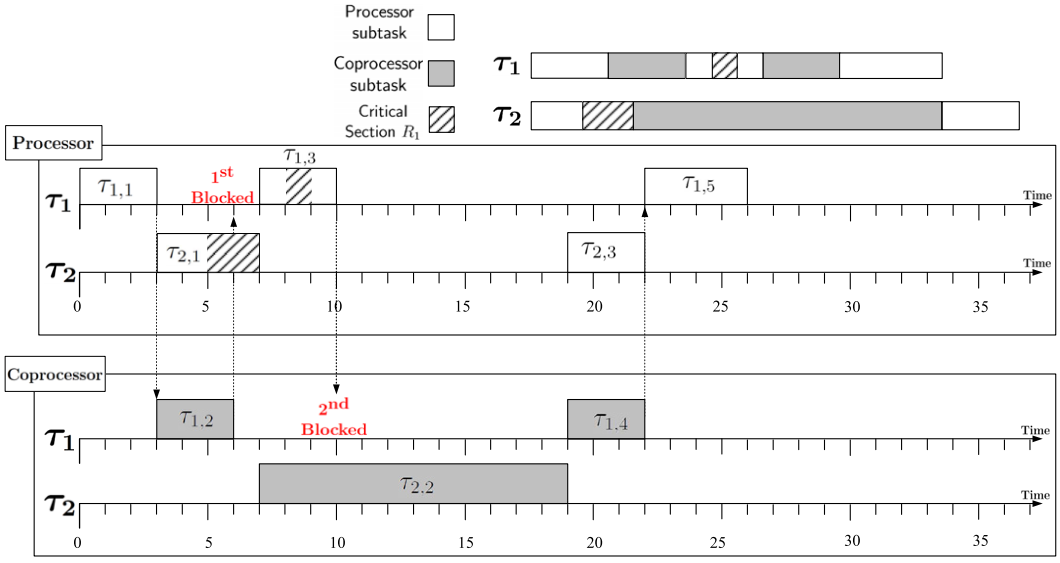
V Frequency switching:
- The processor (coprocessor) frequency is switched to a proper one when a subtask is scheduled or finished its execution at this time point.
- The frequency of the processor (coprocessor) is set based on the following three conditions:
  - The total processor density and total bandwidth server size are updated when there is a new task ready or there are no ready tasks.
  - When the subtask is scheduled without being blocked, the processor (coprocessor) frequency is set as the low frequency $f_{\mathbf{p}}^{\mathbf{L}}$ ($f_{\mathbf{cop}}^{\mathbf{L}}$) according to the total processor density (total bandwidth server size).
  - Whenever a subtask is blocked from starting, the processor (coprocessor) frequency is set as high frequency $f_{\mathbf{p}}^{\mathbf{H}}$ ($f_{\mathbf{cop}}^{\mathbf{H}}$) according to the total processor density (bandwidth server size) with blocking utilization taken into account.

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004

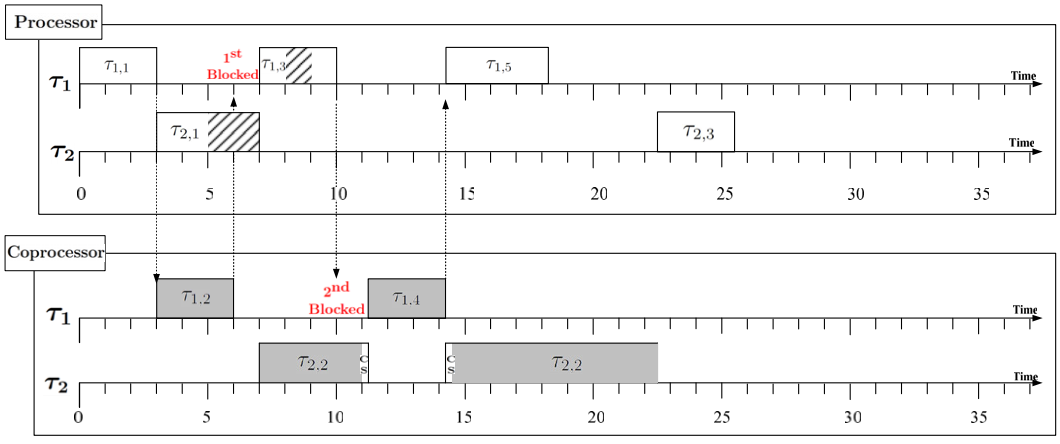
To manage resource usage, the rules of resource allocation and deadline inheritance are the same as in the stack resource policy (SRP) [35]. The task with a shorter deadline is assigned the higher preemption level. All subtasks inherit the preemption level of the corresponding task. Each subtask is scheduled into the ready queue only if its preemption level is higher than $\Pi(t)$, which $\Pi(t)$ denotes the maximum preemption ceiling of resource currently locked by tasks other than task $\tau_i$ at time $t$. The preemption ceiling of a resource is the highest preemption level of all tasks that require the resource. If the preemption level is no higher than $\Pi(t)$, the subtask is blocked. A subtask is blocked only before it is executed under the SRP. The request for a resource is always granted, and the value of $\Pi(t)$ is updated accordingly. When a subtask is blocked, the blocking subtask inherits the shortest deadline among the blocked ones.

As shown in Fig. 2, let us assume that, given a task set $\mathbf{T} = \{\tau_1, \tau_2\}$, task $\tau_1$ and $\tau_2$ have five subtasks and three subtasks, respectively. Both tasks have arrived at time 0. Subtasks $\tau_{1,1}, \tau_{1,3}, \tau_{1,5}, \tau_{2,1}$ and $\tau_{2,3}$ are executed on the processor. Subtasks $\tau_{1,2}, \tau_{1,4}$, and $\tau_{2,2}$ are executed on coprocessor. The periods of $\tau_1$ and $\tau_2$ are 50 and 450 time units, respectively. When the processor operates at its maximum frequency $f_{\mathbf{p}}^{\max}$, the total execution cycles of $\tau_1$'s and $\tau_2$'s processor subtasks are 10 and 7 cycles. Subtasks $\tau_{1,3}$ and $\tau_{2,1}$ require resource $R_1$ for 1 and 2 execution cycles, respectively. When the coprocessor operates at its maximum frequency $f_{\mathbf{cop}}^{\max}$, the total execution cycles of $\tau_1$'s and $\tau_2$'s coprocessor subtasks are 6 and 12 cycles. The preemption point interval *PPI* is 2 cycles and the context switch overhead is 0.25 cycles, when the coprocessor operates at $f_{\mathbf{cop}}^{\max}$.

When we manage two tasks under earliest deadline first (EDF) scheduling with the SRP, as shown in Fig. 2(a), at time 6, $\tau_{1,2}$ finishes and invokes $\tau_{1,3}$, and then $\tau_{1,3}$ is blocked by $\tau_{2,1}$ for accessing resource $R_1$. At time 10, $\tau_{1,3}$ finishes and invokes $\tau_{1,4}$, and then $\tau_{1,4}$ is blocked by $\tau_{2,2}$ for 9 time units from non-preemptible execution on the coprocessor. Let us assume that the processor densities of $\tau_1$ and $\tau_2$ are 0.56 and 0.04, and the bandwidths of $\tau_1$ and $\tau_2$ are 0.24 and 0.06. When the deadline assignment based on a total bandwidth server [36] is used, the deadline of $\tau_{1,1}$ is $0 + \frac{3}{0.56} = 5.4$, the deadline of $\tau_{2,1}$ is $0 + \frac{4}{0.04} = 100$, and the deadline of $\tau_{1,3}$ is $\max\{d_{1,1}, a_{1,3}\} + \frac{3}{0.56} = 10.8$. According to the task scheduling as shown in Algorithm 1, $\tau_{1,3}$ is blocked by $\tau_{2,1}$ for accessing resource $R_1$. At the coprocessor, the deadline of $\tau_{1,2}$ is $a_{1,2} + \frac{3}{0.24} = 15.5$, the deadline of $\tau_{2,2}$ is $7 + \frac{12}{0.06} = 207$, and the deadline of $\tau_{1,4}$ is 28.




(a) EDF+SRP




(b) DCS+SRP

**Fig. 2.** Non-preemptive versus preemptive points.

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004

When Algorithm 1 is used, as shown in Fig. 2(b), $\tau_{1,4}$ preempts $\tau_{2,2}$ at the preemption point, so that the blocking time suffered by task $\tau_{1,4}$ is 1.25 time units from the preemption point interval and the context switch overhead.

As shown in Algorithm 1, the frequency of the processor (coprocessor) is switched when a subtask is scheduled or finished. Given a task set, all subtasks on each core are executed at two alternative frequencies, as presented in prior work [21]. Each subtask is executed at the low frequency $f_{\mathbf{p}}^{\mathbf{L}}$ ($f_{\mathbf{cop}}^{\mathbf{L}}$) on the processor (coprocessor), when it executes without being blocked. Otherwise, the subtask is executed at the high frequency $f_{\mathbf{p}}^{\mathbf{H}}$ ($f_{\mathbf{cop}}^{\mathbf{H}}$) on the processor (coprocessor) with blocking time taken into consideration. Unlike in prior work [21], the frequency $f_{\mathbf{p}}^{\mathbf{L}}$ ($f_{\mathbf{cop}}^{\mathbf{L}}$) is calculated according to the current total processor density (total bandwidth server size). When there is a new task $\tau_i$ added into the system, the processor density $S_i^{\mathbf{p}}$ and a bandwidth server size $S_i^{\mathbf{cop}}$ are assigned according to *EER*. The task is accepted only when it has passed admission control. The required frequencies of each core are then updated according to the processor density and the bandwidth server size of the new task.

Consider the same task set as shown in Fig. 2, let us assume that $f_{\mathbf{p}}^{\max}$ and $f_{\mathbf{cop}}^{\max}$ are normalized to 1. When dual speed (DS) [21] is used, the low and high frequencies are $\sum \frac{c_i}{D_i} = \frac{16}{50} + \frac{19}{450} \approx 0.4$ and $\frac{B_k}{D_k} + \sum \frac{c_i}{D_i} = \frac{12}{50} + \sum \frac{c_i}{D_i} \approx 0.6$, respectively. As shown in Fig. 3(a), without blocking, $\tau_{1,1}$ and $\tau_{1,2}$ are executed at the low frequency. At time 15, $\tau_{1,2}$ finishes and invokes $\tau_{1,3}$, and then $\tau_{1,3}$ is blocked by $\tau_{2,1}$ for resource access. The processor switches to a high frequency to execute the remaining critical section and executes $\tau_{1,3}$. When subtask $\tau_{1,3}$ is finished, $\tau_{1,4}$ is invoked and is then blocked by $\tau_{2,2}$. The coprocessor switches to a high frequency to execute the remaining executions of $\tau_{2,2}$ and $\tau_{1,4}$.

In contrast to DS, under EHDS, the processor and coprocessor subtasks are scheduled by separate schedulers. The high frequency and low frequency are both varied at the processor and the coprocessor. According to the total processor density and total bandwidth server size, the low frequencies of processor $f_{\mathbf{p}}^{\mathbf{L}}$ and coprocessor $f_{\mathbf{cop}}^{\mathbf{L}}$ are $\sum S_i^{\mathbf{p}} \approx 0.6$ and $\sum S_i^{\mathbf{cop}} \approx 0.3$, respectively. When the bandwidth server is used, the non-preemption section effect for bandwidth servers is $\frac{b_{\max}(np)}{\min\{c_{i,j}/S_i^{\mathbf{p}}\}}$ as in Lemma 3.5 [37], where $b_{\max}(np)$ is the longest non-preemption section. The high frequencies of processor $f_{\mathbf{p}}^{\mathbf{H}}$ and coprocessor $f_{\mathbf{cop}}^{\mathbf{H}}$ are $\frac{B}{\min\{c_{i,j}/S_i^{\mathbf{p}}\}} + \sum S_i^{\mathbf{p}} \approx 1$ and $\frac{\max\{PPI,B\}}{\min\{c_{i,j}/S_i^{\mathbf{cop}}\}} + \sum S_i^{\mathbf{cop}} \approx 0.5$, respectively, where *PPI* is the maximum length of the preemption point interval on the coprocessor and $B$ is the maximum length of a subtask that can be blocked for resource access. The proof of the above calculation is shown in Section 3.4. With a lower frequency $f_{\mathbf{cop}}^{\mathbf{L}}$ applied at the coprocessor, $\tau_{1,2}$ executes for 10 time units, and so $\tau_{1,3}$ is not blocked by $\tau_{2,1}$. All processor subtasks are executed at the low frequency. At time 20, when subtask $\tau_{1,3}$ is finished, $\tau_{1,4}$ is invoked and blocked by $\tau_{2,2}$. The coprocessor switches to the high frequency to execute the remaining execution until it reaches the preemption point at $\tau_{2,2}$, whereby it then executes the context switch operation and $\tau_{1,4}$. After $\tau_{1,4}$ finishes, without blocking, the copro-

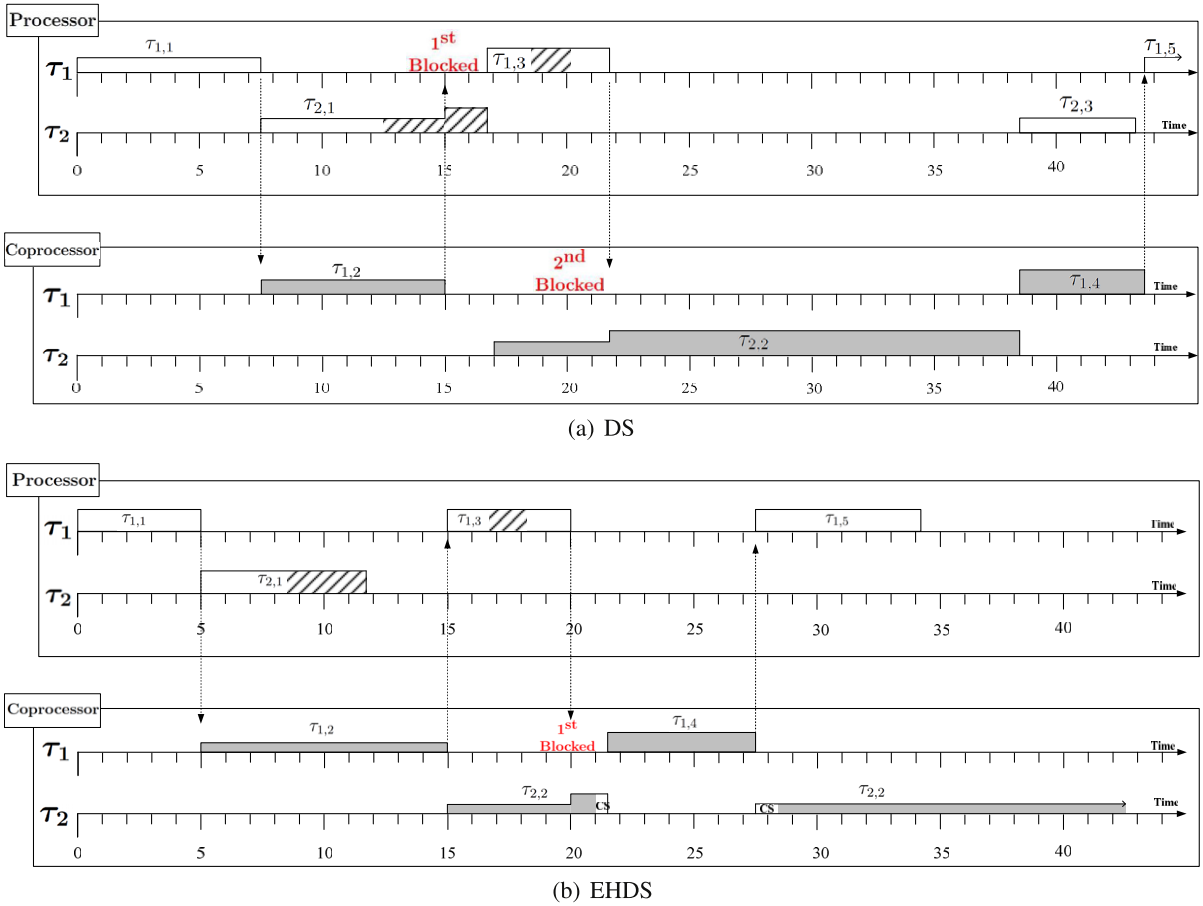



**Fig. 3.** Frequency scaling.

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004

cessor switches to the low frequency to execute the context switch operation and the remaining execution of $\tau_{2,2}$. As shown in this example, EHDS minimizes the system energy by reducing the blocking effect and assigning varied frequencies between cores. In a later section, we explain how to assign frequency and reserve bandwidth to minimize the energy and meet the timing constraint.

### 3.3. Energy efficiency ratio

In this section, we discuss how to assign processor densities and server sizes to each task while considering the different properties of the cores. The assignment of processor densities and server sizes to each task is an NP-complete problem and can be reduced to a bin-packing problem [38]. It should be noted that assigning a larger server size to the task can improve task response, but a smaller server size improves system utilization (further discussion can be found in [13]). In contrast to homogeneous multi-core systems, in a heterogeneous dual-core system, the processor and coprocessor have a master–slave relationship. Due to the precedence constraint, the lower frequency execution on the processor might result in higher frequency execution on the coprocessor for the end-to-end deadline constraint. Moreover, with the same frequency, the resulting energy consumption might be different between the cores due to the varied power characteristics. This section then presents the *energy efficiency ratio EER* for configuring the system energy consumption.

We first define the ratio of total processor subtask utilization to the total coprocessor subtask utilization with preemption point insertion as the following function, where *CS* is the number of context switch cycles of each subtask, and *PPI* is the maximum length of the preemption point interval on the coprocessor:

$$U_{\mathbf{p}} : U_{\mathbf{cop}} = \sum_{\tau_{i,j}\in\mathbf{p}} \frac{c_{i,j}}{D_i - \sum_{\tau_{i,k}\in\mathbf{cop}} PPI} : \sum_{\tau_{i,k}\in\mathbf{cop}} \frac{2CS + c_{i,k}}{D_i - \sum_{\tau_{i,k}\in\mathbf{cop}} PPI}$$

We then partition the deadline to each subtask according to the utilization ratio, schedulability bound, and the energy efficiency ratio *EER* to configure the trade-off between energy consumption and system usage. The context switch from executing a coprocessor subtask is also considered. When the SRP is used, context switching for each subtask only occurs twice [35]. Proportional deadline partition with total utilization [39] is used to reflect the system utilization. As a result, the corresponding density and server size can be calculated by using the functions described below. For each task $\tau_i$,

$$S_i^{\mathbf{p}} = \frac{c_i^{\mathbf{p}}}{(D_i - \sum_{\tau_{i,k}\in\mathbf{cop}} PPI) \times \frac{U_{\mathbf{p}}}{U_{\mathbf{p}}+(EER)\times U_{\mathbf{cop}}} \times U(n)}$$

$$S_i^{\mathbf{cop}} = \frac{2CS + c_i^{\mathbf{cop}}}{(D_i - \sum_{\tau_{i,k}\in\mathbf{cop}} PPI) \times \frac{(EER)\times U_{\mathbf{cop}}}{U_{\mathbf{p}}+(EER)\times U_{\mathbf{cop}}} \times U(n)}$$

For a given task set, the total density and server size of the processor and the coprocessor are $\sum_{i=1}^{n} S_i^{\mathbf{p}} = \frac{U_{\mathbf{p}}+(EER)\times U_{\mathbf{cop}}}{U(n)}$ and $\sum_{i=1}^{n} S_i^{\mathbf{cop}} = \frac{U_{\mathbf{p}}+(EER)\times U_{\mathbf{cop}}}{EER\times U(n)}$, respectively. To meet the deadline constraint, the required frequencies to execute tasks on the processor and the coprocessor are no less than the total bandwidth, i.e. $f_{\mathbf{p}}^{\mathbf{L}} \geqslant \sum_{i=1}^{n} S_i^{\mathbf{p}}$ and $f_{\mathbf{cop}}^{\mathbf{L}} \geqslant \sum_{i=1}^{n} S_i^{\mathbf{cop}}$. The detailed proof is shown in Section 3.4. By assigning the ratio of the total bandwidth between the processor and the coprocessor as *EER*, the frequency ratio between cores can also be *EER*, where *EER* represents the energy efficiency ratio to adjust the energy consumptions contributed from processor and coprocessor.

Next, we discuss how to calculate the energy efficiency ratio *EER* to minimize the energy consumption. Given a task set, the total energy of the system is as follows:

$$\begin{aligned}
E_{\mathbf{system}} &= E_{\mathbf{p}} + E_{\mathbf{cop}} \\
&= \left\{ \left[ \sum_{i=1}^{n} P_{\mathbf{p}}(f_{\mathbf{p}}) \times \frac{c_i^{\mathbf{p}}}{f_{\mathbf{p}}} \times \frac{HP}{D_i} \right] + \beta_{\mathbf{p}} \times HP \right\} \\
&\quad + \left\{ \left[ \sum_{i=1}^{n} P_{\mathbf{cop}}(f_{\mathbf{cop}}) \times \frac{c_i^{\mathbf{cop}}}{f_{\mathbf{cop}}} \times \frac{HP}{D_i} \right] + \beta_{\mathbf{cop}} \times HP \right\} \\
&= HP \times \left\{ P_{\mathbf{p}}(f_{\mathbf{p}}) \times \frac{1}{f_{\mathbf{p}}} \times \sum_{i=1}^{n} \frac{c_i^{\mathbf{p}}}{D_i} + P_{\mathbf{cop}}(f_{\mathbf{cop}}) \times \frac{1}{f_{\mathbf{cop}}} \times \sum_{i=1}^{n} \frac{c_i^{\mathbf{cop}}}{D_i} + \beta_{\mathbf{p}} + \beta_{\mathbf{cop}} \right\} \\
&\approx HP \times \left\{ \left(\alpha_{\mathbf{p}} f_{\mathbf{p}}^3\right) \times \frac{1}{f_{\mathbf{p}}} \times U_{\mathbf{p}} + \left(\alpha_{\mathbf{cop}} f_{\mathbf{cop}}^3\right) \times \frac{1}{f_{\mathbf{cop}}} \times U_{\mathbf{cop}} + \beta_{\mathbf{p}} + \beta_{\mathbf{cop}} \right\}
\end{aligned}$$

where $HP$ is the hyper-period of the task set, $f_{\mathbf{p}}$ is the current frequency of the processor, $c_i^{\mathbf{p}}$ is the execution time on the processor, $D_i$ is the relative deadline of the task $\tau_i$, $\beta_{\mathbf{p}}$ is the leakage power of the processor, and $U_{\mathbf{p}}$ is the total processor subtask utilization. The above variables with subscript **cop** are the values corresponding to those characteristics of the coprocessor.

As the hyper-period *HP* of a task set and the leakage power of each core are constants, the minimization goal is transformed as

$$\alpha_{\mathbf{p}} f_{\mathbf{p}}^2 U_{\mathbf{p}} + \alpha_{\mathbf{cop}} f_{\mathbf{cop}}^2 U_{\mathbf{cop}}$$

with the schedulability bound constraint:

$$\sum_{i=1}^{n} \frac{c_i^{\mathbf{p}}}{D_i \times f_{\mathbf{p}}} + \sum_{i=1}^{n} \frac{c_i^{\mathbf{cop}}}{D_i \times f_{\mathbf{cop}}} \approx \frac{U_{\mathbf{p}}}{f_{\mathbf{p}}} + \frac{U_{\mathbf{cop}}}{f_{\mathbf{cop}}} \leqslant U(n)$$

As shown in above equation, the value of *EER* is affected by the power characteristics of the cores and schedulability utilization of the given scheduler. This optimization problem can be resolved by the Lagrange multiplier [40]. The energy efficiency ratio, i.e., $f_{\mathbf{p}} : f_{\mathbf{cop}}$, is derived as $\sqrt[3]{\alpha_{\mathbf{cop}}/\alpha_{\mathbf{p}}}$. Without considering blocking, the optimal $f_{\mathbf{cop}}^{\mathbf{L}}$ and $f_{\mathbf{p}}^{\mathbf{L}}$ are equal to $\frac{U_{\mathbf{p}}+(EER)\times U_{\mathbf{cop}}}{U(n)}$ and $\frac{U_{\mathbf{p}}+(EER)\times U_{\mathbf{cop}}}{EER\times U(n)}$ as the results of the above assignments.

The energy caused by run-time blocking is unpredictable and further examination might lead us into probability problem [11]. In this study, we try to minimize the energy caused by run-time blocking through preemption point insertion as shown in Fig. 3. In this example, let us assume that $\alpha_{\mathbf{p}}$ and $\alpha_{\mathbf{cop}}$ are 1 and 8, and that the value of *EER* is $\sqrt[3]{\alpha_{\mathbf{cop}}/\alpha_{\mathbf{p}}} = 2$. Without considering power leakage, the total energies of the task set (during hyper-period 450) of EHDS and DS are 82 and 100 energy units, respectively. With run-time blocking considered, the total energies of EHDS and DS are 87 and 120 energy units, respectively.

Moreover, the run-time idle slacks caused by high frequency execution can be reclaimed to minimize the effects of blocking. The main ideas follow the dual speed dynamic reclaiming (DSDR) proposed in [21]. Two separate free-time lists record the run-time slacks of processor subtasks and coprocessor subtasks, respectively, but the slacks are consumed when the core is idle. The slacks of a subtask on the free-time list expire by the deadline of the corresponding subtask. Each subtask can only use the slacks of subtasks with shorter relative deadlines. More experimental results for this are shown in Section 4.

The idea of this paper is to propose a configurable framework for a heterogeneous dual-core system to make a trade-off between low system energy and high system utilization. If there is no way to assign the bandwidth ratio between cores as *EER* under the timing constraint, then the system energy will be increased. Developers can repartition the workload between cores to minimize the system energy. To address the system utilization issue, the sufficient schedulability bound is 1 under EHDS. With the precedence constraint, the bound is varied between task sets [37]. Developers can perform a binary search to explore the acceptable bound in this

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004

configurable framework, while the system robustness is guaranteed by the admission control theorems shown in the next section.

### 3.4. Admission control

In this section, we present the polynomial time schedulability test and frequency assignment for fast admission control. Many bandwidth server techniques are designed for dynamic scheduling algorithms. To compare with the EDF scheduling with SRP, this study uses a total bandwidth server (TBS) [36] with EDF scheduling as an example to demonstrate how to schedule tasks with the precedence constraint and preemption points. The admission control is based on the concept of TBS to simplify the presentation. When the processor and coprocessor are executed at full speed, that is, $f_{\mathbf{p}}^{\mathbf{L}} = f_{\mathbf{p}}^{\max}$ and $f_{\mathbf{cop}}^{\mathbf{L}} = f_{\mathbf{cop}}^{\max}$, the schedulability of EHDS follows the concept proposed by Chen [13] as follows:

**Theorem 3.1.** *[13] Given a task set with a processor density assignment, if the sum of the processor density $\sum_{i=1}^{n} S_i^{\mathbf{p}}$ is no larger than 1, then the task set is schedulable on the processor under dual-core scheduling.*

**Theorem 3.2.** *[13] Given a task set if, $\forall i,j$ $\frac{PPI}{\min\left\{\frac{c_{i,j}}{S_i^{\mathbf{cop}}}\right\}} + \sum_{i=1}^{n} S_i^{\mathbf{cop}} \leqslant 1$, the task set is schedulable on the coprocessor under dual-core scheduling, where PPI is the maximum length of the preemption point interval and $S_i^{\mathbf{cop}}$ is the bandwidth server size of task $\tau_i$ on the coprocessor.*

**Theorem 3.3.** *[13] A task $\tau_i$ is schedulable under dual-core scheduling if, $\forall j,k$ $\sum_{\tau_{i,j}\in\mathbf{p}} \frac{c_{i,j}}{S_i^{\mathbf{p}}} + \sum_{\tau_{i,k}\in\mathbf{cop}} \left(\frac{2CS+c_{i,k}}{S_i^{\mathbf{cop}}} + PPI\right) \leqslant D_i$, where CS is the number of context switch cycles of each subtask, PPI is the maximum length of the preemption point interval on the coprocessor, $S_i^{\mathbf{p}}$ is the processor density of task $\tau_i$, $S_i^{\mathbf{cop}}$ is the bandwidth server size of task $\tau_i$, and $D_i$ is the relative deadline of task $\tau_i$.*

When dual-core scheduling is applied, each subtask is scheduled by two separate EDF schedulers with bandwidth reservation on the processor and the coprocessor. With the SRP and an EDF scheduler on a processor, two alternative frequencies to minimize energy without deadline violation can be calculated as follows:

**Theorem 3.4.** *[21] Suppose that n periodic tasks with blocking sections are sorted by their periods. They can all be feasibly scheduled by the dual speed EDF algorithm with high frequency $f^{\mathbf{H}}$ and low frequency $f^{\mathbf{L}}$ if, $\forall k$ $\frac{B_k}{D_k} + \sum_{i=1}^{n} \frac{c_i}{D_i} \leqslant f^{\mathbf{H}}$ and $\sum_{i=1}^{n} \frac{c_i}{D_i} \leqslant f^{\mathbf{L}}$, where $B_k$ is the maximum length of a job in $\tau_k$ that can be blocked.*

All subtasks on a processor are scheduled by the assigned processor density and hence, the dual speed concept can be applied directly to the processor after taking into account processor density. The low frequency is not less than the total processor density, and the high frequency is derived from the total processor density including blocking effects. When bandwidth reservation is used, the non-preemption critical section effect for bandwidth servers is $\frac{b_{\max}(np)}{\min\{c_{i,j}/S_i^{\mathbf{p}}\}}$ [37], as shown in Lemma 3.5.

**Lemma 3.5.** *[37] When a system of periodic tasks is scheduled with one or more total bandwidth servers on the an EDF basis, every periodic task and every server meets its deadlines if the sum of the total density of the periodic tasks and the total size of all servers is no greater than $1 - \frac{b_{\max}(np)}{D_{\min}}$, where $b_{\max}(np)$ and $D_{\min}$ are the maximum execution time of the non-preemption portion and minimum of the relative deadlines of all periodic tasks and the effective execution times of jobs executed by all servers in the system, respectively.*

From Theorem 3.4 and Lemma 3.5, the schedulability under EHDS of a processor can be derived as Corollary 3.6.

**Corollary 3.6.** *Given a task set with the processor density assignment and blocking sections, the task set is schedulable on the processor under EHDS with low frequency $f_{\mathbf{p}}^{\mathbf{L}}$ and high frequency $f_{\mathbf{p}}^{\mathbf{H}}$ if,*

$$\sum_{i=1}^{n} S_i^{\mathbf{p}} \leqslant f_{\mathbf{p}}^{\mathbf{L}}$$

*and*

$$\forall j,k \quad \frac{B_k}{\min\left\{\frac{c_{i,j}}{S_i^{\mathbf{p}}}\right\}} + \sum_{i=1}^{n} S_i^{\mathbf{p}} \leqslant f_{\mathbf{p}}^{\mathbf{H}},$$

*where $B_k$ is the maximum length of a subtask in $\tau_k$ that can be blocked.*

**Proof.** The proof of this corollary directly follows from that of Theorem 3.1, Theorem 3.4 and Lemma 3.5. As discussed previously, the low frequency of the coprocessor is no less than the total bandwidth server size. The high frequency is derived with worst case blocking assumed, including resource contention and the non-preemption execution portion, because subtasks are scheduled by the assigned bandwidth server at the preemption point. □

**Corollary 3.7.** *Given a task set with the bandwidth server size assignment and blocking sections, the task set is schedulable on the coprocessor under EHDS with low frequency $f_{\mathbf{cop}}^{\mathbf{L}}$ and high frequency $f_{\mathbf{cop}}^{\mathbf{H}}$ if,*

$$\sum_{i=1}^{n} S_i^{\mathbf{cop}} \leqslant f_{\mathbf{cop}}^{\mathbf{L}}$$

*and*

$$\forall j,k \quad \frac{\max\{PPI, B_k\}}{\min\left\{\frac{c_{i,j}}{S_i^{\mathbf{cop}}}\right\}} + \sum_{i=1}^{n} S_i^{\mathbf{cop}} \leqslant f_{\mathbf{cop}}^{\mathbf{H}},$$

*where PPI is the maximum length of the preemption point interval on the coprocessor and $B_k$ is the maximum length of a subtask in $\tau_k$ can be blocked for resource contention.*

**Proof.** The proof of this corollary directly follows from that of Theorem 3.2, Theorem 3.4 and Lemma 3.5. □

**Theorem 3.8.** *A task $\tau_i$ is schedulable under EHDS if,*

$$\forall j,k, \quad \sum_{\tau_{i,j}\in\mathbf{p}} \frac{c_{i,j}}{S_i^{\mathbf{p}}} + \sum_{\tau_{i,k}\in\mathbf{cop}} \left(\frac{2CS+c_{i,k}}{S_i^{\mathbf{cop}}} + PPI\right) \leqslant D_i \tag{1}$$

$$\forall \tau_{i,j} \in \mathbf{p}, \quad \frac{c_{i,j}}{f_{\mathbf{p}}^{\mathbf{H}}} + \frac{B_{i,j}}{f_{\mathbf{p}}^{\mathbf{H}}} \leqslant \frac{c_{i,j}}{S_i^{\mathbf{p}}} \tag{2}$$

*and*

$$\forall \tau_{i,k} \in \mathbf{cop}, \quad \frac{2CS+c_{i,k}}{f_{\mathbf{cop}}^{\mathbf{H}}} + \frac{\max\{PPI, B_{i,k}\}}{f_{\mathbf{cop}}^{\mathbf{H}}} \leqslant \frac{c_{i,k}}{S_i^{\mathbf{cop}}}, \tag{3}$$

*where CS is the number of context switch cycles of each subtask on the coprocessor, $B_{i,j}$ is the maximum length of a subtask $\tau_{i,j}$ can be blocked by resource usage, and PPI is the maximum length of the preemption point interval on the coprocessor.*

**Proof.** According to Corollary 3.6 and Corollary 3.7, the low frequency of the processor, $f_{\mathbf{p}}^{\mathbf{L}}$, or the coprocessor, $f_{\mathbf{cop}}^{\mathbf{L}}$, is no less than the bandwidth of each task $\tau_i$. Following Theorem 3.3, Eq. (1) guar-

antees that response time of task $\tau_i$ is no larger than its deadline $D_i$. According to the properties of SRP [35], each subtask is blocked only once before it begins execution. When a subtask of task $\tau_i$ is blocked by the resource contention ($B_{i,j}$), the processor is set as the high frequency until the blocking section and the blocked subtask are finished. The coprocessor is set as the high frequency, when a subtask of task $\tau_i$ is blocked by the non-preemption portion (i.e., *PPI*) or by the resource contention ($B_{i,k}$). Eq. (2) and Eq. (3) guarantee that the required execution time units of each subtask $\tau_{i,j}$ are no larger than the local relative deadline of the corresponding subtask $\tau_{i,j}$.

Considering the task set and frequency assignment as shown in Fig. 3, for task $\tau_1$, it meets the end-to-end deadline constraint of Eq. (1), $\sum_{\tau_{i,j}\in \mathbf{p}} \frac{c_{i,j}}{S_i^{\mathbf{p}}} + \sum_{\tau_{i,k}\in \mathbf{cop}} \left(\frac{2CS+c_{i,k}}{S_i^{\mathbf{cop}}} + PPI\right) = \frac{10}{0.56} + \frac{6}{0.24} + 2 \approx 45 \leqslant 50$, each processor subtask meets the conditions of Eq. (2), (e.g., for $\tau_{1,1}$ $\left(\frac{3}{1} + \frac{2}{1}\right) \leqslant \frac{3}{0.56} \approx 5$), and each coprocessor subtask meets the conditions of Eq. (3), (e.g., for $\tau_{1,2}$ $\left(\frac{2\times 0.25+3}{0.5} + \frac{2}{0.5}\right) \leqslant \frac{3}{0.24} = 12.5$). Task $\tau_2$ is also schedulable by passing the above conditions. □

## 4. Performance evaluation

### 4.1. Data sets and performance metrics

The purpose of this section is to evaluate energy consumption in task execution. We compare the power consumption of our EHDS algorithm, DS, and SRP, which executes all tasks for the maximum frequency with the EDF scheduler. The specifications of an OMAP35x Evaluation Module [12] containing an ARM Cortex A9 for the processor and a C64X + DSP for the coprocessor, are used for this evaluation, in which the power consumption can be approximately modeled as $P(f) = (0.0013f^3 + 40)$ W for ARM and $P(f) = (0.0026f^3 + 41)$ W for DSP. The frequency levels of the OMAP35x Evaluation Module are shown in Table 1. Without the specialized claim, tasks are executed by any frequency in the given range, in which the maximum and minimum available frequencies are normalized to 1 and 0.1, respectively. The effect of discrete frequency is also evaluated in Section 4.2.5. The workloads in the experiments are generated in a randomized way. The utilization of the system ranges from 0.1 to 0.9. The number of tasks in each core is 10. The generated periods of each task ranges from 200 to 1300 cycles. The ratio of total processor subtask utilization to the total coprocessor subtask utilization is 2. The execution cycles of each task are given with the harmonic utilization. Each task has 3–7 subtasks and each subtask is executed with the processor and coprocessor interleaved. The number of resources ranges from 2 to 6. The length of resource usage is from 20% to 66% of the execution time in each subtask. For each setting, 100 task sets are generated to average the experimental results.

The primary metric is the *normalized energy consumption*. Let $X$ and $Y$ denote the amount of energy consumption under the evaluated algorithm and the amount of energy consumption under the SRP, respectively. The *normalized energy consumption* of the evaluated algorithm is defined as $X/Y$. The second metric is *infeasibility* which is the ratio of the number of task set with deadline violation under the evaluated algorithm to the number of total task sets generated.

**Table 1**
Frequency levels of OMAP 35x.

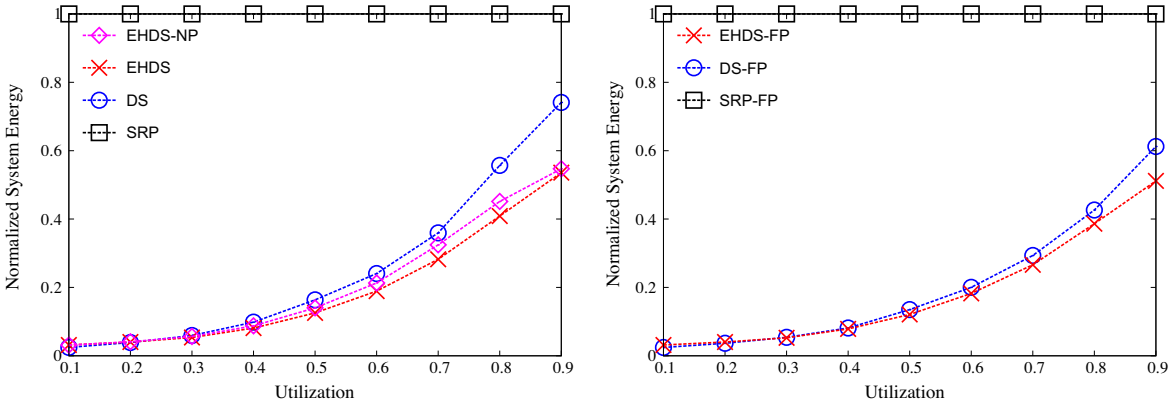
| Core | Frequency | Normalized speed |
|---|---|---|
| ARM | 125 | $0.22\ f_{\mathbf{p}}^{\max}$ |
| | 250 | $0.45\ f_{\mathbf{p}}^{\max}$ |
| | 500 | $0.90\ f_{\mathbf{p}}^{\max}$ |
| | 550 | $1\ f_{\mathbf{p}}^{\max}$ |
| DSP | 90 | $0.20\ f_{\mathbf{cop}}^{\max}$ |
| | 180 | $0.41\ f_{\mathbf{cop}}^{\max}$ |
| | 400 | $0.93\ f_{\mathbf{cop}}^{\max}$ |
| | 430 | $1\ f_{\mathbf{cop}}^{\max}$ |

### 4.2. Experiment results

In this section, we present the results from experiments using different workloads to compare our EHDS algorithm against DS and SRP.

#### 4.2.1. Varying utilization

Fig. 4(a) shows the normalized energy consumptions of EHDS, DS, EHDS-NP, and SRP. EHDS schedules a coprocessor subtask at the preemption point, and the length of *PPI* is equal to the maximum length of shared resource usage. DS, SRP, and EHDS-NP execute non-preemptible tasks on the coprocessor. The horizontal axis represents the system utilization, i.e., $\sum \frac{c_i}{D_i}$, and the vertical axis represents the normalized energy consumption. The performance is better when the energy consumption is less. The results show that EHDS achieved the lowest normalized energy consumption compared to the other algorithms. It is because the bandwidth reservation through energy efficiency ratio improves the system utilization and hence minimizes the energy. Furthermore, the




(a) Non-preemptive coprocessor (b) Fully preemptive coprocessor

**Fig. 4.** Varying utilizations.

preemption point insertion decreases the blocking time and eliminates the energy from high speed execution. Compared with DS and SRP, EHDS saves up to 20% and 45% more energy, respectively. The performance of EHDS-NP is slightly worse than EHDS, but still better than DS because of the energy efficiency ratio assignment. Fig. 4(b) shows the normalized energy consumption of EHDS-FP, DS-FP, and SRP-FP, and all task executions on the coprocessor can be fully preemptible. The performance gap is decreased by preemptible execution, but EHDS-FP still performs better than DS-FP.

#### 4.2.2. Varying preemption point interval

Fig. 5(a and b) shows the normalized energy consumptions of EHDS, DS, and SRP, when the system utilizations are 0.8 and 0.9, respectively. The horizontal axis represents the length of the *PPI*, and the vertical axis represents the normalized energy consumption. The maximum length of *PPI* is set as 70, because the coprocessor subtask almost becomes non-preemptible by such setting. All algorithms schedule subtasks on the coprocessor only at the preemption point. As shown in Fig. 5, the energy of all algorithms increases with the length of the *PPI*. However, the performance gap of EHDS is small because the bandwidth reservation takes the value of *PPI* into account. It should be noted that the system utilization is less than 1 in order to compare with DS and SRP. In a later section, we show the experimental results for schedulability capability of EHDS.

#### 4.2.3. Schedulability

Fig. 6 shows the schedulability capabilities of DS and EHDS. The horizontal axis represents system utilization, and the vertical axis represents infeasibility. In this experiment, 100 task sets were generated for different system utilizations without schedulability tests. As shown in Fig. 6, the schedulability bound of EHDS is closed to 1.3. This is because the bandwidth server assignment as shown in Section 3.3 takes system utilization into account to achieve better results.

#### 4.2.4. Run-time reclaiming

Fig. 7 shows the normalized energy consumptions of algorithms with and without run-time reclaiming. The horizontal axis represents the system utilization, and the vertical axis represents the normalized energy consumption. In this experiment, we extend DS-DR [21] as two separate lists for the processor and the coprocessor to reclaim the energy from the blocking effect. EHDS-DR and DS-DR reclaim run-time slacks from the blocking effect. As shown in Fig. 7, the performance of EHDS is still better than that of DS-DR. Compared with EHDS, the EHDS-DR saves up to 10% more energy.

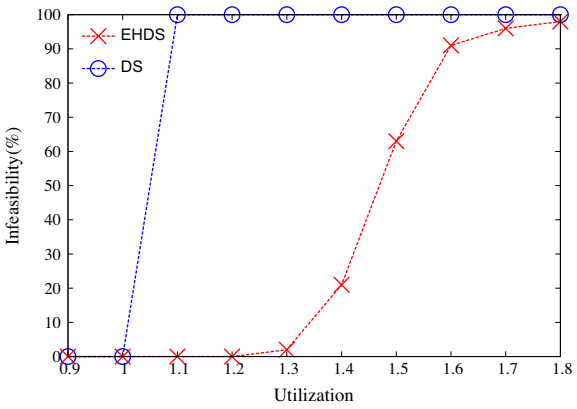



**Fig. 6.** Schedulability.

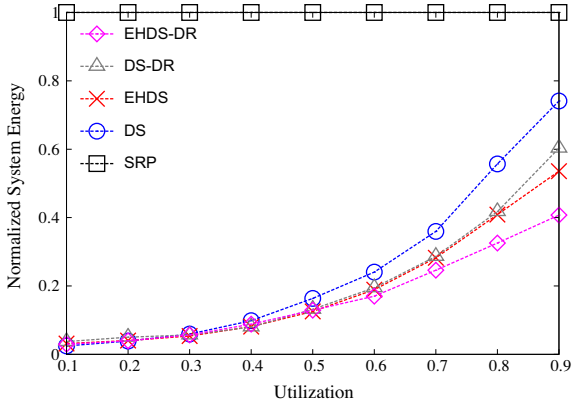



**Fig. 7.** Reclaiming.

#### 4.2.5. Discrete frequency

Fig. 8 shows a comparison of discrete and continuous frequency levels, when the discrete frequency levels of the OMAP35x Evaluation Modules are used. As shown in Fig. 8, EHDS-DF and DS-DF shows the energy consumption when the discrete frequency levels are applied. The effects of the discrete frequency levels of the cores results in significant energy increases when $U = 0.5$. The reason for this is the large gap between the second and third frequency levels of the cores, as shown in Table 1. As a result, as compared to continuous frequency levels, the energy increases up to 30% due to the effects of discrete frequency. The performance of EHDS-DF is still

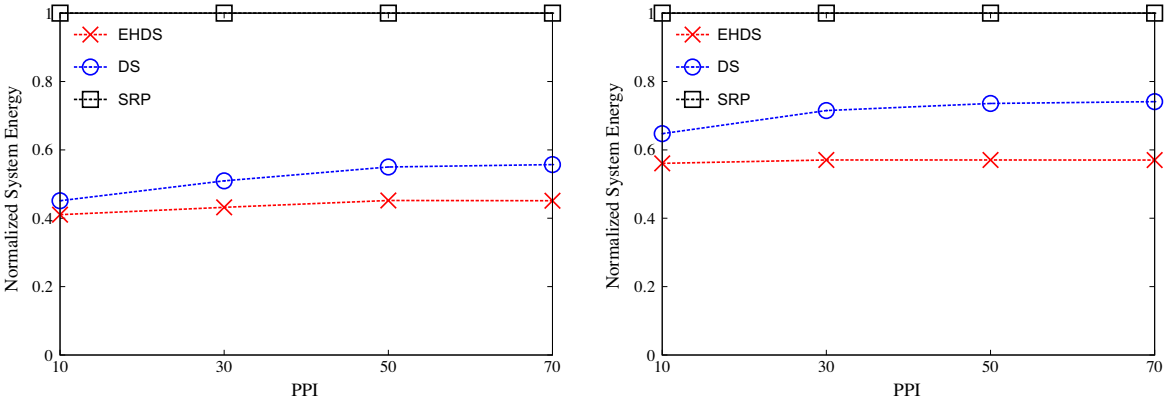



**Fig. 5.** Varying preemption point intervals.

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004

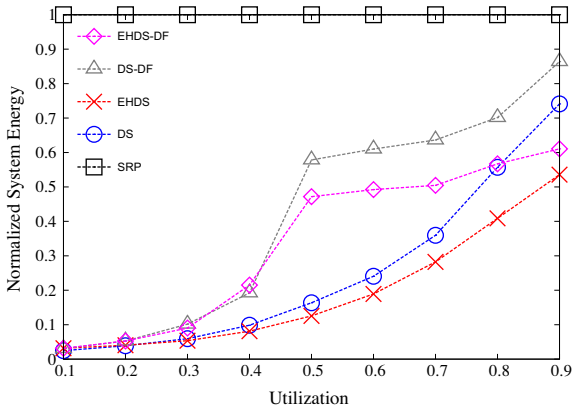



**Fig. 8.** Discrete frequency.

better than that of DS-DF because the lower frequency level is assigned to the coprocessor through the energy efficiency ratio, and then this eliminates the effect from discrete frequency.

## 5. Conclusion

This paper explores the on-line energy-efficient real-time task scheduling problem in heterogeneous dual-core systems, and considers tasks with precedence constraints and non-preemptible execution. The precedence constraint and non-preemptive coprocessor are dealt with through the bandwidth server scheduling framework and preemption point insertion proposed in [13]. To ensure energy efficiency in this scheduling framework, fast frequency assignment is proposed to accept a dynamic workload and minimize system energy. The energy efficiency ratio is presented to configure low energy consumption and high system utilization. A heuristic approach for derivation of the energy efficiency ratio is also presented to provide further insights into workload partitioning between cores. The capabilities of the proposed algorithms are evaluated by a series of experiments over synthesized workloads. Performance evaluations show that our proposed algorithms can reduce the energy consumption and outperform previous approaches.

Future research should extend this framework to heterogeneous multi-core systems and derive the utilization bounds of such systems. Further research and development in these areas may also prove to be very important for future mobile system designs.

## Acknowledgement

The authors would like to thank Hung-Hsiu Tsai at the National Taiwan University of Science and Technology for his contribution to part of the experimental design. This work is supported in part by a grant from the NSC program NSC 99-2221-E-011-116-MY3.

## References

[1] Texas Instruments Inc., OMAP3 Platform, Tech. Rep., Texas Instruments, 2009. Available from: <http://www.ti.com/lit/ml/swpt024b/swpt024b.pdf>.
[2] Texas Instruments Inc., OMAP Platform, Tech. Rep., Texas Instruments, 2011. Available from: <http://focus.ti.com/omap/docs/omaphomepage.tsp>.
[3] Qualcomm Inc., Snapdragon, Tech. Rep., Qualcomm, 2011. Available from: <http://www.qualcomm.com/documents/snapdragon-s4-architecture>.
[4] J.-J. Chen, H.-R. Hsu, T.-W. Kuo, Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems, in: Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, 2006, pp. 408–417.
[5] J.-J. Chen, L. Thiele, Energy-efficient scheduling on homogeneous multiprocessor platforms, in: Proceedings of the ACM Symposium on Applied Computing, 2010, pp. 542–549.
[6] H. Aydin, R. Melhem, D. Mosse', P. Meji'a-Alvarez, Dynamic and aggressive scheduling techniques for power-aware real-time systems, in: Proceedings of the IEEE Real-Time Systems Symposium, 2001, pp. 95–105.
[7] X. Qi, D. Zhu, H. Aydin, Global reliability-aware power management for multiprocessor real-time systems, in: Proceedings of the Embedded and Real-Time Computing Systems and Applications, 2010, pp. 183–192.
[8] H.K. Xuefeng Piao, Y. Cho, S. Han, M. Park, M. Park, S. Cho, Power-aware edzl scheduling upon identical multiprocessor platforms, in: Proceedings of the Reliable and Autonomous Computational Science International Conference, 2010, pp. 61–80.
[9] H. Aydin, Q. Yang, Energy-aware partitioning for multiprocessor real-time systems, in: Proceedings of the International Parallel and Distributed Processing Symposium, 2003, p. 9.
[10] R. Xu, R. Melhem, D. Mosse, Energy-aware scheduling for streaming applications on chip multiprocessors, in: Proceedings of the Real-Time Systems Symposium, 2007, pp. 25–38.
[11] A. Abousamra, R. Melhem, D. Mosse, Minimizing expected energy consumption for streaming applications with linear dependencies on chip multiprocessors, in: Proceedings of the IEEE International Symposium on Industrial Embedded Systems, 2009, pp. 100–109.
[12] W.-Y. Shieh, B.-W. Chen, Energy-efficient tasks scheduling algorithm for dual-core real-time systems, in: Proceedings of the International Computer Symposium, 2010, pp. 568–575.
[13] Y.-S. Chen, L.-P. Chang, C.-M. Jeng, On-line task scheduling for dual-core real-time embedded systems, in: Proceedings of the Conference on Industrial Informatics, 2009, pp. 182–187.
[14] C.-F. Kuo, Y.-C. Hai, Real time task scheduling on heterogeneous two-processor systems, in: Proceedings of the Conference on Algorithms and Architectures for Parallel Processing, 2010, pp. 68–78.
[15] P. Gai, L. Abeni, G. Buttazzo, Multiprocessor dsp scheduling in system-on-a-chip architecture, in: Proceedings of the Euromicro Conference on Real Time Systems, 2002, pp. 231–238.
[16] K. Kim, D. Kim, C. Park, Real-time scheduling in heterogeneous dual-core architecture, in: Proceedings of the Conference on Parallel and Distributed Systems, 2006, p. 6.
[17] S. Kato, K. Lakshmanan, R. Rajkumar, Y. Ishikawa, Timegraph: GPU scheduling for real-time multi-tasking environments, in: Proceedings of the USENIX Annual Technical Conference, 2011, pp. 17–30.
[18] S. Kato, K. Lakshmanan, Y. Ishikawa, R. Rajkumar, Resource sharing in GPU-accelerated window systems, in: Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, 2011, pp. 191–200.
[19] Y. Wang, D. Liu, M. Wang, Z. Qin, Z. Shao, Optimal task scheduling by removing inter-core communication overhead for streaming applications on MPSoC, in: Proceedings of the Real-Time and Embedded Technology and Applications Symposium, 2010, pp. 195–204.
[20] Y. Wang, D. Liu, Z. Qin, Z. Shao, Memory-aware optimal scheduling with communication overhead minimization for streaming applications on chip multiprocessors, in: Proceedings of the Real-Time Systems Symposium, 2010, pp. 350–359.
[21] F. Zhang, S.T. Chanson, Processor voltage scheduling for real-time tasks with non-preemptible sections, in: Proceedings of the Real-Time Systems Symposium, 2002, pp. 235–245.
[22] J. Lee, K. Koh, C.-G. Lee, Multi-speed DVS algorithms for periodic tasks with non-preemptible sections, in: Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2007, pp. 459–468.
[23] A. Elewi, M. Awadalla, M. Eladawy, Energy-efficient multi-speed algorithm for scheduling dependent real-time tasks, in: Proceedings of the International Conference on Computer Engineering Systems, 2008, pp. 237–242.
[24] R. Jejuilar, R. Gutpa, Energy aware task scheudling with task synchoronization for embedded real-time system, IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems 25 (6) (2006) 1024–1037.
[25] Y.-S. Chen, C.-Y. Yang, T.-W. Kuo, Energy-efficient task synchronization for real-time systems, IEEE Transactions on Industrial Informatics 6 (3) (2010) 287–301.
[26] D. Zhu, Reliability-aware dynamic energy management in dependable embedded real-time systems, in: Proceedings of the Real-Time and Embedded Technology and Applications Symposium, 2006, pp. 397–407.
[27] R. Jejurikar, C. Pereira, R.K. Gupta, Leakage aware dynamic voltage scaling for real-time embedded systems, in: Proceedings of the Design Automation Conference, 2004, pp. 275–280.
[28] J.-J. Chen, T.-W. Kuo, C.-S. Shih, $1+\epsilon$ approximation clock rate assignment for periodic real-time tasks on a voltage-scaling processor, in: Proceedings of the ACM Conference on Embedded Software, 2005, pp. 247–250.
[29] W.-C. Kwon, T. Kim, Optimal voltage allocation techniques for dynamically variable voltage processors, in: Proceedings of the Design Automation Conference, 2003, pp. 125–130.
[30] J. Rosen, P. Eles, Z. Peng, A. Andrei, Predictable worst-case execution time analysis for multiprocessor systems-on-chip, in: Proceedings of the IEEE International Symposium on Electronic Design, Test and Application, 2011, pp. 99–104.
[31] L. Steffens, M. Agarwal, P. van der Wolf, Real-time analysis for memory access in media processing SOCS: a practical approach, in: Proceedings of the Euromicro Conference on Real-Time Systems, 2008, pp. 255–265.
[32] M. Bautin, A. Dwarakinath, T. Chiueh, Graphics engine resource management, in: Proceedings of Annual Multimedia Computing and Networking Conference, 2008, p. 12 pp.
[33] Texas Instruments Inc., DSP/BIOS II Timing Benchmarks on the TMS320C54x DSP, Tech. Rep., Texas Instruments, 2000. <http://focus.ti.com>.

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004

[34] K.-Y. Hsieh, Y.-C. Lin, C.-C. Huang, J.-K. Lee, Enhancing microkernel performance on VLIW DSP processors via multiset context switch, Journal of Signal Processing Systems 51 (3) (2008) 257–268.
[35] T.P. Baker, Stack-based resource allocation policy for real-time process, in: Proceedings of the Real Time Systems Symposium, 1990, pp. 191–200.
[36] M. Spuri, G. Buttazo, Scheduling aperiodic tasks in dynamic priority systems, Real-Time Systems 10 (2) (1996) 179–210.
[37] J.W. Liu, Real-Time systems, Prentice Hall, 2000.
[38] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
[39] B. Kao, H. Garcia-Molina, Deadline assignment in a distributed soft real-time system, IEEE Transactions on Parallel and Distributed Systems 8 (12) (1997) 1268–1274.
[40] J. Stewart, Calculus: Early Transcendentals, Brooks Cole, 2002.

**Ya-Shu Chen** joined Department of Electrical Engineering, National Taiwan University of Science and Technology, at August 2007. She currently serves as an Assistant Professor. Ya-Shu Chen earned her BS degree in computer information and science at National Chiao-Tung University in 2001. Then, she studied in Department of Computer Science and Information Engineering, National Taiwan University, and was supervised by Prof. Tei-Wei Kuo. She successfully defended her master thesis and doctoral dissertation at 2003 and 2007, respectively. Her research interest includes operating systems, embedded systems, and hardware/software co-design.

**Ming-Yang Chen** joined Multisuns Corporation, at March 2012. He currently serves as a Software Development Engineer. He received his M.S. degree in electrical engineering from National Taiwan University of Science and Technology at 2012, and was supervised by Dr. Ya-Shu Chen. He earned his B.S. degree in electronic engineering from Fu Jen Catholic University at 2008. His research interests include operating systems, and energy-efficient real-time scheduling in heterogeneous multi-core systems.

Please cite this article in press as: Y.-S. Chen, M.-Y. Chen, On-line energy-efficient real-time task scheduling for a heterogeneous dual-core system-on-a-chip, J. Syst. Architect. (2012), http://dx.doi.org/10.1016/j.sysarc.2012.05.004