Energy Efficiency Analysis for the Single Frequency Approximation (SFA) Scheme

SANTIAGO PAGANI and JIAN-JIA CHEN, Karlsruhe Institute of Technology (KIT)

Energy-efficient designs are important issues in computing systems. This article studies the energy efficiency of a simple and linear-time strategy, called the Single Frequency Approximation (SFA) scheme, for periodic real-time tasks on multicore systems with a shared supply voltage in a voltage island. The strategy executes all the cores at a single frequency to just meet the timing constraints. SFA has been adopted in the literature after task partitioning, but the worst-case performance of SFA in terms of energy consumption incurred is an open problem. We provide comprehensive analysis for SFA to derive the cycle utilization distribution for its worst-case behaviour for energy minimization. Our analysis shows that the energy consumption incurred by using SFA for task execution is at most 1.53 (1.74, 2.10, 2.69, respectively), compared to the energy consumption of the optimal voltage/frequency scaling, when the dynamic power consumption is a cubic function of the frequency and the voltage island has up to 4 (8, 16, 32, respectively) cores. The analysis shows that SFA is indeed an effective scheme under practical settings, even though it is not optimal. Furthermore, since all the cores run at a single frequency and no frequency alignment for Dynamic Voltage and Frequency Scaling (DVFS) between cores is needed, any unicore dynamic power management technique for reducing the energy consumption for idling can be easily incorporated individually on each core in the voltage island. This article also provides an analysis of energy consumption for SFA combined with procrastination for Dynamic Power Management (DPM), resulting in an increment of 1 from the previous results for task execution. Furthermore, we also extend our analysis for deriving the approximation factor of SFA for a multicore system with multiple voltage islands.

Categories and Subject Descriptors: D.4.7 [**Operating Systems**]: Organization and Design—*Real-time systems and embedded systems*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Single frequency approximation, SFA, energy efficiency, voltage island, power management, multicore

ACM Reference Format:
Santiago Pagani and Jian-Jia Chen. 2014. Energy efficiency analysis for the single frequency approximation (SFA) scheme. ACM Trans. Embedd. Comput. Syst. 13, 5s, Article 158 (September 2014), 25 pages.
DOI: http://dx.doi.org/10.1145/2660490

1. INTRODUCTION

Energy-efficient and low-power designs have become important issues for computing systems in order to prolong the battery lifetime of embedded systems, or to reduce the power bills for servers. This is one of the main motivations for why single-core computing systems have moved to multicore platforms, mainly to balance the power consumption and computation performance.

As shown in the literature, such as Jejurikar et al. [2004], the dynamic power consumption (mainly generated by switching activities) and the static power consumption

This work is supported in part by Baden Wurttemberg MWK Juniorprofessoren-Programme.
Authors' addresses: S. Pagani (corresponding author), Department of Informatics, Karlsruhe Institute of Technology (KIT), Haid-und-Neu-Str. 7, 76131 Karlsruhe, Germany; email: santiago.pagani@kit.edu; J.-J. Chen, Department of Informatics, TU Dortmund University, Otto-Hahn-Str. 16, 44227 Dortmund, Germany.

DOI: http://dx.doi.org/10.1145/2660490

(mainly generated by the leakage current) are the two major sources of power consumption in CMOS processors. When static power is negligible, due to the convexity of the power consumption function, it is usually better to execute at lower frequency for energy minimization by using the Dynamic Voltage and Frequency Scaling (DVFS) technique. However, for systems with nonnegligible static power, the energy consumption function for execution is no longer an increasing function. Hence, executing any task at some frequency lower than a *critical frequency* might consume more energy for execution, since the static power plays a role. This motivates the combination of DVFS and Dynamic Power Management (DPM), so that cores can be slowed down and then shut down after finishing their workloads (e.g., [Irani et al. 2003; Albers and Antoniadis 2012; Jejurikar et al. 2004]).

In the past decade, task scheduling and partitioning have been explored for energy reduction while still meeting the performance requirements. However, most research either assumes that each core can change its own supply voltage independently from the others (denoted as per-core DVFS, e.g., [Chen et al. 2006; Aydin and Yang 2003; Xu et al. 2005; de Langen and Juurlink 2006; Chen and Thiele 2010; Moreno and de Niz 2012]), or considers the other extreme and energy-inefficient direction where there exists only one shared supply voltage whereby the maximum frequency of all cores is limited (denoted as global DVFS), such as Yang et al. [2005], Devadas and Aydin [2010], and Seo et al. [2008].

For next-generation many-core systems, a trade-off between global DVFS and per-core DVFS platforms is to adopt a multicore architecture with different voltage islands in which the cores on a voltage island share the same supply voltage (modifiable by DVFS at any time), but individual islands can have different voltages [Borkar 2007; Herbert and Marculescu 2007]. For example, Intel has released a research multicore platform called Single-chip Cloud Computer (SCC) [Intel 2009; Howard et al. 2011] with such a feature. The cores on a voltage island are naturally consolidated as a cluster.

Related Work. For per-core DVFS, power-aware and energy-efficient scheduling for homogeneous multicore systems has been widely explored, especially for real-time embedded systems (e.g., [Chen et al. 2006; Aydin and Yang 2003; Xu et al. 2005; de Langen and Juurlink 2006; Chen and Thiele 2010; Moreno and de Niz 2012]). It has been shown in Chen and Thiele [2010] that applying the Largest-Task-First (LTF) strategy for task mapping results in solutions with approximation factors, in terms of energy consumption, in which the factors depend on the hardware platforms. Specifically, by turning off a processor to reduce the energy consumption in homogeneous multiprocessor systems, Xu et al. [2005] and Chen and Thiele [2010] propose polynomial-time algorithms to derive task mappings that try to execute at a *critical frequency*. For homogeneous multiprocessor systems with discrete voltage levels and frequencies, de Langen and Juurlink [2006] provide heuristics for energy-aware scheduling, and Moreno and de Niz [2012] present an algorithm that runs in polynomial time and computes the optimal voltage and frequency assignment for systems with uniform frequency steps and negligible static/leakage power consumption. Providing an individual supply voltage for each core locally can be energy efficient but is costly for implementation. Based on VLSI circuit simulations, it has been suggested in Herbert and Marculescu [2007] that per-core DVFS suffers complicated design problems.

For global voltage scaling, the result in Yang et al. [2005] provides answers on voltage scaling to minimize the energy consumption by using an accelerating schedule when the system has *frame-based real-time tasks* (all the tasks have the same arrival time and period). However, the approach in Yang et al. [2005] is highly restricted and cannot be easily extended to handle periodic real-time tasks (where tasks have

different arrival times and periodicity), or systems with nonnegligible static/leakage power consumption. The study in Devadas and Aydin [2010] and Seo et al. [2008] relaxes the assumptions in Yang et al. [2005] by considering periodic real-time tasks with nonnegligible static- and voltage-independent power consumptions and nonnegligible overhead for turning to low-power idle modes. The approach in Devadas and Aydin [2010] decides the number of active cores and then the frequency of the active cores. However, there is no theoretical analysis in Devadas and Aydin [2010] to show the effectiveness of their approach for energy minimization. The work in Seo et al. [2008] dynamically balances the task loads of multiple cores to optimize power consumption during execution and adjusts the number of active cores to reduce leakage power consumption under low-load conditions.

Motivation. When considering energy efficiency for scheduling periodic real-time tasks on multicore systems with a shared supply voltage in a voltage island (or a system with a global supply voltage), it is necessary to choose a policy that decides the voltage of the island and the frequencies of the cores for execution. The simplest and most intuitive strategy is to use a single voltage and frequency for executing, particularly the lowest voltage and frequency that satisfies the timing constraints. We denote such a scheme as the Single Frequency Approximation (*SFA*) scheme. After the task partitioning is done (which is not the focus of this article), SFA has linear-time complexity, only from evaluating the core with the highest cycle utilization.

Even though SFA is not an optimal strategy for energy efficiency, it significantly reduces the management overhead. SFA does not require frequent voltage/frequency changes at runtime, as it only requires one voltage and one frequency. Furthermore, since no frequency alignment for DVFS between cores is needed under SFA, any unicore Dynamic Power Management (DPM) technique can be adopted individually in each core, together with SFA, with no additional effort.

SFA has been adopted by several researchers in the past, for instance, Devadas and Aydin [2010] (when tasks do not complete earlier than the estimated worst-case execution times) and Nikitin and Cortadella [2012]. SFA is indeed a good strategy when the workload is *perfectly* balanced, that is, when all the cores are assigned with the same cycle utilization. On the contrary, if the utilizations of the cores are *skewed*, that is, one core with high cycle utilization and all the others with very low cycle utilization, then SFA would consume much more energy than the optimal solution, especially when the number of cores in the voltage island grows. This comes from the cases wherein cores with light cycle utilizations are forced to run at higher frequencies than they need to meet their timing constraints.

Therefore, we know that SFA is a practical approach for executing periodic tasks after task partitioning in multicore systems in a voltage island. Moreover, under such settings, we are also not aware of any other good heuristic voltage/frequency scheduling algorithms that have such low overhead, low energy consumption, and that allow for easy integration with existing DPM techniques. However, the worst-case performance of SFA, in terms of energy consumption, is an open problem.

Objective. Motivated by the preceding discussions, the goal of this article is to provide comprehensive analysis from a theoretical point of view to show the effectiveness of SFA for energy minimization, particularly for state-of-the-art designs that have a limited number of cores per voltage island.

Our Contributions. Under fixed task sets of periodic real-time tasks in a voltage island, our contributions are as follows.

—We reveal the effectiveness of SFA for energy efficiency and show that it has an approximation factor (worst-case ratio of the energy consumption for SFA against

the optimal energy consumption) that can be bounded, in which the factor depends on the parameters of the power consumption function and the number of cores per voltage island.
—Furthermore, we also show the effectiveness of SFA by analysing the approximation factor when applying SFA together with the unicore DPM procrastination scheme from Irani et al. [2003] and with the optimal DPM solution [Baptiste et al. 2012]. We show that the overall approximation factor by considering such DPM schemes and SFA is increased by 1 from the analysis in Section 6.
—Specifically, we evaluate several cases with practical settings when the number of cores in the voltage island is limited. When the voltage island has up to 4 (8, 16, 32, respectively) cores, the approximation factor of SFA for minimizing the energy consumption for execution is at most 1.53 (1.74, 2.10, 2.69, respectively). The factor can be further improved if the task sets are balanced.
—Finally, we sketch simple extensions to consider cores with a limited set of available frequencies and systems with multiple voltage islands.

The analysis in this article shows that SFA is an effective scheme which dramatically simplifies the complexity compared to current DVFS algorithms, even though not optimal. We believe that the analysis for SFA for fixed task sets (that are already mapped onto cores) can be used as the cornerstone when considering task partitioning.

Organization. The rest of the article is organized as follows.

—Section 2 details the system model and defines the problem to solve.
—In Section 3, we derive a lower bound for the energy consumption of the cores in a voltage island for periodic real-time tasks.
—Section 4 provides the energy consumption of the cores in a voltage island that uses SFA to decide the voltage of the island and the frequencies of the cores.
—In Section 5 we derive the approximation factor of SFA in terms of energy consumption when the static/leakage power consumption is considered negligible.
—Continuing the analysis, Section 6 derives the approximation factor of SFA in terms of energy consumption when the static/leakage power consumption is nonnegligible.
—Section 7 constrains the analysis for the case wherein the system uses a load balancer for task partitioning, thus deriving a lower value for the approximation factor under such conditions.
—On the other hand, Section 8 extends the analysis in the previous sections to consider nonnegligible energy overhead for entering/leaving a low-power mode.
—Moreover, Section 9 presents numerical results for the theoretical worst-case approximation factors derived in Section 6, Section 7, and Section 8.
—Section 10 presents an extension to consider systems with discrete frequencies.
—Section 11 extends all the previous analysis, which is for a single voltage island, to consider multicore systems with multiple voltage islands.
—In Section 12 we simulate the performance of SFA in terms of energy consumption for different scenarios and in a single voltage island. Thus, we provide a *concrete factor* based on the energy consumption of SFA.
—Finally, Section 14 concludes the article.

2. SYSTEM MODEL AND PROBLEM DEFINITION

This section reviews the power and energy model adopted for the rest of the article and defines the problem to solve.

2.1. Hardware Model

This article focuses on a single voltage island, where all the cores in the island have the same supply voltage and run at the same frequency at any given time point, such as one voltage island of SCC [Intel 2009; Howard et al. 2011].[1] The system can change the voltage and frequency of the island by adopting DVFS. This model has also been adopted in Yang et al. [2005] and Devadas and Aydin [2010]. For a core to support a frequency, the supply voltage in the island has to be adjusted accordingly, in particular to the least available supply voltage such that stable execution on the core is achievable for the frequency. The available frequencies are in the range of $[s_{\min}, s_{\max}]$.[2]

We denote the power consumption of a core executing a certain task at frequency s as $P(s)$. The energy consumption during time interval Δt at frequency s is denoted as $E(s) = P(s) \cdot \Delta t$. We assume that $P(s)$ is a convex and increasing function with respect to s, which complies with most of the power models for CMOS processors adopted in the literature, for example, Aydin and Yang [2003], Xu et al. [2005], de Langen and Juurlink [2006], Yang et al. [2005], and Devadas and Aydin [2010]. Among these, the most widely used power consumption function, adopted for this article, is

$$P(s) = \beta + \alpha s^{\gamma}, \tag{1}$$

where $\alpha > 0$ is a constant dependent on the effective switching capacitance, $\gamma > 1$ is related to the hardware, and $\beta \geq 0$ represents the static power consumption.

During interval Δt, a core running at frequency s executes a certain amount Δc of core cycles such that $\Delta t = \frac{\Delta c}{s}$ and

$$E(s) = (\beta + \alpha s^{\gamma}) \frac{\Delta c}{s}. \tag{2}$$

This energy consumption is a convex function. Hence, by setting to zero the first-order derivative of Eq. (2) with respect to s, the minimum value for $E(s)$ is found when s is $\sqrt[\gamma]{\frac{\beta}{(\gamma-1)\alpha}}$. In order to consider the case when such value is smaller than $s_{\min}$, we define the critical frequency as $s_{\text{crit}} = \max\{s_{\min}, \sqrt[\gamma]{\frac{\beta}{(\gamma-1)\alpha}}\}$. The critical frequency represents the frequency that minimizes the energy consumption for execution when the overhead for sleeping is considered negligible, as also shown in Jejurikar et al. [2004] and Chen et al. [2006].

Furthermore, we can use the power consumption function from Eq. (1) to model the experimental results from Howard et al. [2011], in which a multicore system that integrates 48 cores was developed. Figure 12 (frequency versus voltage) and Figure 13 (measured power versus voltage) from Howard et al. [2011] are of particular interest and the values are summarized in Figure 1(a) and Figure 1(b).

We approximate the table in Figure 1(a) using a quadratic function. Thus, by having a function that relates frequency with voltage, we are able to rewrite the table in Figure 1(b) relating frequency with power. The power values of this new table are divided by 48, since these experiments were conducted on the entire chip, but we are interested in the power function of each individual core. Finally, we approximate this new table with the power consumption function from Eq. (1), where $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\gamma = 3$, and $\beta = 0.5$ Watts, resulting in $s_{\text{crit}} = 0.52$ GHz. These values result in a goodness of fit of Sum of Squares due to Error (SSE) of 0.05041, a Square of the correlation between the response values and the predicted response values (*R-square*) of 0.9958,

[1]The analysis will be extended for multiple voltage islands in Section 11.

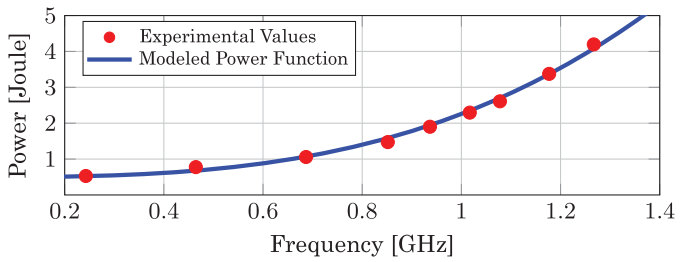
[2]For systems with discrete frequencies, all the analysis still holds based on a simple extension presented in Section 10.

Voltage	Frequency
0.73 V	301.48 MHz
0.75 V	368.82 MHz
0.85 V	569.45 MHz
0.94 V	742.96 MHz
1.04 V	908.92 MHz
1.14 V	1077.11 MHz
1.23 V	1223.37 MHz
1.32 V	1303.79 MHz

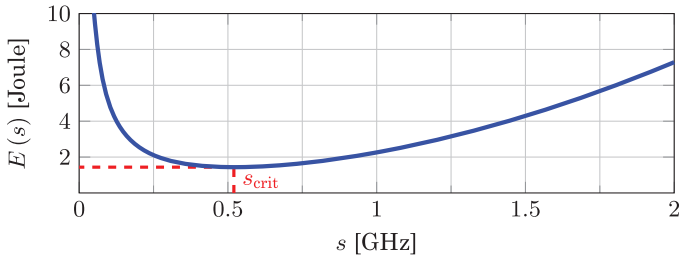
(a) frequency vs. voltage

Voltage	Total Power
0.70 V	25.38 W
0.80 V	37.26 W
0.91 V	50.76 W
1.00 V	70.73 W
1.05 V	91.25 W
1.10 V	110.15 W
1.14 V	125.27 W
1.21 V	161.99 W
1.28 V	201.40 W

(b) power vs. voltage



(c) power model for a single core



(d) energy consumption for a single core executing 10^9 computer cycles

Fig. 1. Experimental results from the 48-core system in Howard et al. [2011] and the derived power model for a single core with $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\gamma = 3$, $\beta = 0.5$ Watts, and $s_{\text{crit}} = 0.52$ GHz.

an adjusted R-square of 0.9958 and a Root Mean Squared Error (RMSE) of 0.07938. Figure 1(c) shows this power model and the original experimental measurements from Howard et al. [2011]. Moreover, Figure 1(d) illustrates function $E(s)$ from Eq. (2) for a single core executing 10^9 computer cycles with the same α, γ, and β values, where it is easy to observe the concept of a critical frequency.

When a core finishes executing all its workload in the ready queue, it has to wait until a new task instance arrives. During this waiting interval, the core can choose to stay idle (in execution mode), consuming β power. Moreover, the core can also choose to go to a low-power mode that consumes $\beta' \geq 0$ power. As we can transfer the power consumption β' to the power consumption of the voltage island for being active, without loss of generality, we can set $P(s)$ as $P(s) - \beta'$ such that we can disregard the effect of the power consumption of a core in a low-power mode. Nevertheless, the core consumes some energy during the transition process of entering/leaving the low-power mode. Given that we handle periodic tasks that always go back to the execution mode after a certain amount of time, we denote the overhead for sleeping as the summation of the energy consumption both for entering and leaving the low-power mode.

When the duration of the waiting interval until the arrival of a new task instance is short enough, idling is more energy efficient than sleeping. Contrarily, when the interval is sufficiently long, sleeping results in higher energy savings. Thus, we define the break-even time as the time such that the energy consumption for idling is equal to the overhead for sleeping.

The analysis for Sections 4 to 6 considers negligible overhead for sleeping, for which the break-even time is 0 and every core goes to sleep immediately when it has no workload to execute. For systems with nonnegligible overhead, the strategy by considering the break-even time and procrastination schemes, such as in Irani et al. [2003] and Chen and Kuo [2007], can be further adopted; this is extended in Section 8.

2.2. Task Model

We consider periodic real-time tasks with implicit deadlines, where each task τ_j releases an infinite number of task instances with period (and relative deadline) p_j and each instance has worst-case execution cycles e_j. We consider partitioned scheduling in which each task is assigned onto a core, that is, when a task instance arrives at the system, it is executed on the assigned core. Specifically, we use *Earliest-Deadline-First* (EDF) scheduling in which the task instance with the earliest absolute deadline on a core has the highest priority. The least common multiple (LCM) of the periods of all tasks is called the *hyperperiod* and is denoted as L.

After the task partitioning is completed by using M cores[3], the tasks are grouped into M task sets $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M\}$. Note that the task partitioning is not the focus of this article and is assumed given. Without loss of generality, we assume that task set $\mathbf{T}_i$ is assigned on core i and we define its *cycle utilization* as $w_i = \sum_{\tau_j \in \mathbf{T}_i} \frac{e_j}{p_j}$. By defining w_0 for simplicity and without loss of generality, we order the cores such that $0 = w_0 \leq w_1 \leq w_2 \leq \cdots \leq w_M$. It has been well studied, for instance, in Liu and Layland [1973], that executing core i at a frequency higher than or equal to w_i with EDF will meet the timing constraint.

2.3. Problem Definition

We consider the Single Frequency Approximation (SFA) scheme in which all cores in the island always execute at single frequency s_u and each core enters a low-power mode after executing its workload. The time complexity of SFA is $O(M)$ to ensure the feasibility, where M is the number of cores in the voltage island and this complexity comes only from evaluating the highest cycle utilization. Clearly, s_u must be at least w_M to ensure feasible schedules.

The objective of this work is to analyse the approximation factor of SFA, defined $\mathrm{AF}_{\mathrm{SFA}}$, and expressed as

$$\mathrm{AF}_{\mathrm{SFA}} = \max \frac{E_{\mathrm{SFA}}}{E_{\mathrm{OPT}}} \leq \max \frac{E_{\mathrm{SFA}}}{E^*}, \tag{3}$$

where E_{OPT} is the optimal energy consumption during a hyperperiod, E_{SFA} is the energy consumption for SFA during a hyperperiod, and E^* is a lower bound for the optimal energy consumption for any feasible schedule during a hyperperiod. Since E_{OPT} is not easily obtained, in the analyses we use its lower bound E^* that should not be very far away from E_{OPT}.

Note that SFA does not require any capability of voltage/frequency scaling at runtime, as it only uses one frequency. However, to explore the approximation factor we need E^*, in which changing the supply voltage and frequency of the island is with negligible overhead and the available frequencies are continuous between $(0, s_{\max}]$. This approach results in a safe lower bound for the optimal energy consumption. We only focus on the analysis of the approximation factor. The applicability of SFA with slack reclamation to deal with early completion of tasks can be found in Devadas and Aydin [2010].

To improve the readability of this article, Figure 2 presents a reference diagram that relates all equations, lemmas, theorems, and figures.

[3]Note that, if the tasks are partitioned into M' task sets with $M' > M$, this would result in an infeasible task partition as there would be more task sets than cores. Furthermore, if the tasks are partitioned into M' task sets with $M' < M$, all the analysis in the following sections still holds by simply considering that the voltage island has M' cores, which in fact reduces the approximation factor.

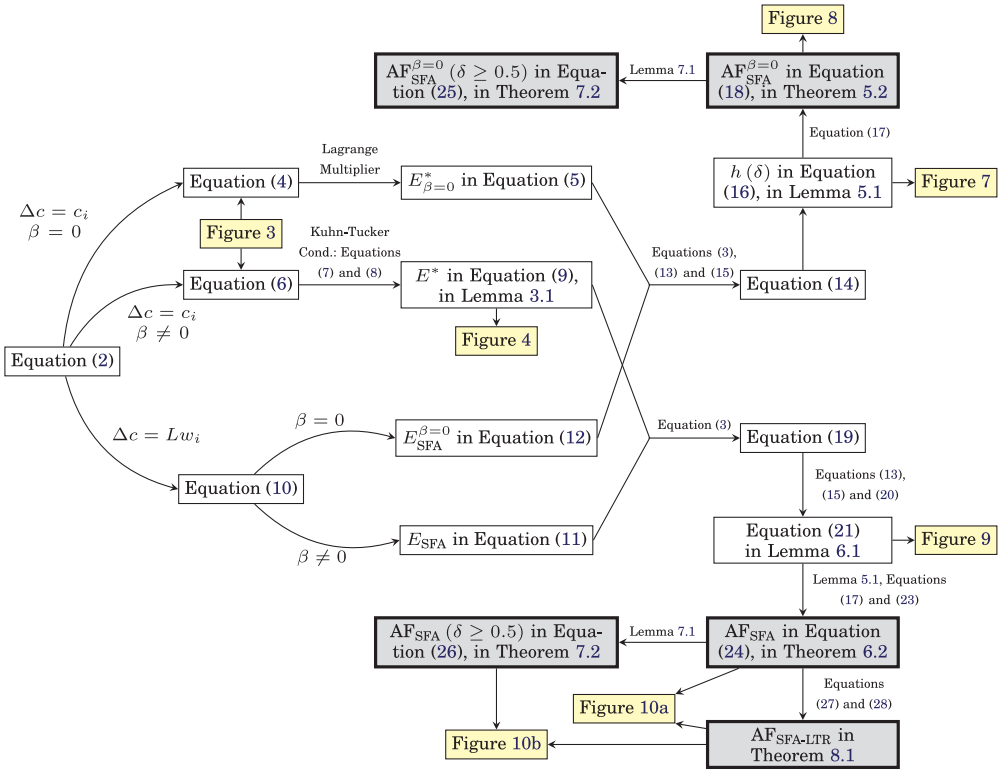


Fig. 2. Reference diagram relating equations, lemmas, theorems, and figures.

3. LOWER BOUND ENERGY CONSUMPTION

This section provides a lower bound for the energy consumption for periodic real-time tasks that is needed to obtain the approximation factor in Eq. (3).

3.1. Preliminary Results with $\beta = 0$

A special case of the problem is with frame-based real-time tasks in which all the tasks arrive at time 0 and have the same period and deadline. For such a case, the period of any task is also the hyperperiod L. Yang et al. [2005] have proposed a scheme based on the *deep sleeping property* (every core is put to sleep after executing its workload), as shown in Figure 3. For completeness, we summarize the schedule proposed in Yang et al. [2005] when the task sets are already assigned onto cores. In Yang et al. [2005], the schedule is divided into M fragments. In the i-th fragment, all cores run at speed s_i during time t_i. Moreover, in the i-th fragment there are $M - i + 1$ cores that execute $c_i = L(w_i - w_{i-1})$ core cycles during t_i such that $t_i = \frac{c_i}{s_i}$ (the rest of the cores are in the sleep state). Therefore, considering Eq. (2) with $\beta = 0$ and $\Delta c = c_i$ for each fragment, the energy consumed by the active cores during t_i in the i-th fragment, with $s_{\min} = 0$, is expressed as

$$E(t_i) = \sum_{i=1}^{M} (M - i + 1)\alpha \frac{c_i^{\gamma}}{t_i^{\gamma}} t_i. \tag{4}$$

By computing the values of t_i based on the Lagrange multiplier method with constraint $\sum_{i=1}^{M} t_i = L$, Yang et al. [2005] provide an optimal frequency assignment for the

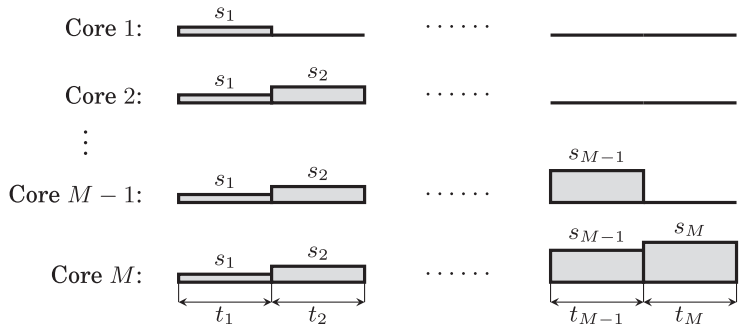


Fig. 3. A schedule satisfying the deep sleeping property.

previous case, that results in an energy consumption of

$$E^*_{\beta=0} = \alpha L \left[\sum_{i=1}^{M} (w_i - w_{i-1}) \sqrt[\gamma]{M-i+1} \right]^{\gamma}. \tag{5}$$

However, when we move further to consider periodic real-time tasks, it becomes very complicated to calculate the optimal frequency scaling to minimize the energy consumption for executing tasks. Fortunately, to analyse the approximation factor, we only need a lower bound of the energy consumption.

For systems with periodic real-time tasks, the workload to be completed on core i during the hyperperiod is $L \cdot w_i$. To derive E^*, we can simply consider that all the $L \cdot w_i$ workload arrives at time 0 and has to be done by time L. Even though an optimal frequency scaling for such a relaxation is not always a feasible solution for periodic tasks, such a simplification provides a good lower bound estimation of the optimal energy consumption. Finally, when $\beta = 0$, we simply apply the result from Yang et al. [2005] in Eq. (5).

Note that this simplification is only applied for obtaining the lower bound. This *may* result in pessimistic analysis, but does not limit the applicability of SFA for the considered task models. Furthermore, Section 9 presents numerical examples that show the analysis is in fact not pessimistic.

3.2. Lower Bound with $\beta \neq 0$

This section analyses the lower bound of the energy consumption when β is not negligible. Similar to Section 3.1, we can again consider the same relaxation when all the tasks arrive at time 0 and negligible overhead for entering/leaving low-power modes and changing the supply voltage of the island, which is a safe approach. For the same schedule as in Yang et al. [2005], with $s_{\min} = 0$ and $\beta \neq 0$, Eq. (4) changes to

$$E(t_i) = \sum_{i=1}^{M} (M-i+1) \left(\beta + \alpha \frac{c_i^{\gamma}}{t_i^{\gamma}} \right) t_i. \tag{6}$$

To obtain the lower bound for energy consumption, we apply the Kuhn-Tucker conditions [Rardin 1998] on Eq. (6) under the constraint $\sum_{i=1}^{M} t_i \leq L$ and $t_i \geq 0$ for $i = 1, 2, \ldots, M$. Due to space constraints, the details are omitted. Once the Lagrangian is solved, the set of t_i that minimizes the energy consumption for frame-based tasks is

$$t_i = \sqrt[\gamma]{\frac{\alpha(\gamma-1)(M-i+1)}{(M-i+1)\beta + \lambda}} c_i. \tag{7}$$

When $\sum_{i=1}^{M} t_i < L$, then λ is 0, and from Eq. (7) the resulting t_i is equal to $\frac{c_i}{s_{\text{crit}}}$ for all $i = 1, 2, \ldots, M$. In other words, all the cores run at frequency s_{crit} for execution. Clearly, when $w_M \leq s_{\text{crit}}$, the preceding solution is a feasible one.

For the case that $\sum_{i=1}^{M} t_i = L$, then $\lambda > 0$, conceptually meaning that it is no longer feasible to meet the timing constraints by running all cores at s_{crit} for execution. Hence, from Eq. (7) it holds that

$$\sum_{i=1}^{M} t_i = \sum_{i=1}^{M} \sqrt[\gamma]{\frac{\alpha\,(\gamma-1)(M-i+1)}{(M-i+1)\,\beta+\lambda}}\, c_i = L. \tag{8}$$

The only unknown variable in Eq. (8) is λ. Since Eq. (8) is strictly decreasing with respect to λ, one possibility to derive it is to apply Newton's method. However, Newton's method only gives the numerical results for a specific case study, but we do not have an explicit form to solve Eq. (8). Therefore, for analysing the worst-case performance for a given task partitioning, we have to find a safe approximation for estimating lower bound E^* for the optimal energy consumption. Lemma 3.1 shows how we estimate E^* based on an auxiliary frequency defined as s_{dyn}.

LEMMA 3.1. *For a given frequency s_{dyn} with $s_{crit} < s_{dyn} < s_{max}$, a safe lower bound for the optimal energy consumption for any feasible schedule can be expressed as*

$$E^*(w_M) = \begin{cases} \alpha\gamma L \left({s_{crit}}^{\gamma-1}\right) \sum_{i=1}^{M} w_i & \text{if } w_M \leq s_{dyn} \\ \alpha L \left[\sum_{i=1}^{M} (w_i - w_{i-1}) \sqrt[\gamma]{M-i+1}\right]^{\gamma} & \text{otherwise.} \end{cases} \tag{9}$$

PROOF. According to the previous analysis and lower bound for the optimal energy consumption in Eq. (5) by ignoring the static and independent power consumption, we know that both cases are safe lower bounds for the energy consumption. Therefore, either one can be adopted.

The goal for using s_{dyn} is to provide a tighter lower bound of energy consumption by choosing these two lower bounds in proper cases. It is clear that, when w_M is high, the energy consumption resulting from the dynamic power plays a more important role. Therefore, for a given s_{dyn}, when $w_M > s_{\text{dyn}}$, we only consider the dynamic energy consumption in Eq. (9). The other case considers the lower bound of energy consumption by running at the critical frequency. □

An example of the lower bound for the energy consumption from Eq. (9) can be found in Figure 4.

4. ENERGY CONSUMPTION OF SFA

This section analyses the energy consumption of SFA needed to obtain the approximation factor in Eq. (3).

For periodic real-time tasks, the workload to be completed on core i during the hyperperiod is $L \cdot w_i$. From Eq. (2), with $\Delta c = L \cdot w_i$, the energy consumption for core i under SFA is $(\beta + \alpha s_u^{\gamma})\frac{w_i}{s_u} L$, therefore, the energy consumption for all M cores in the

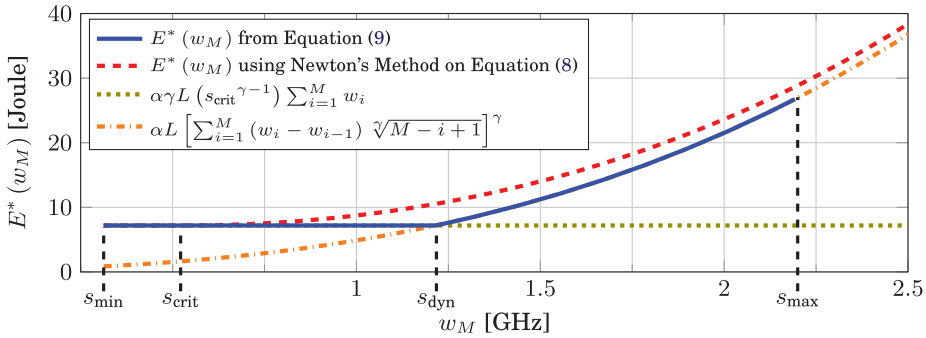


Fig. 4. $E^*(w_M)$ with $\alpha = 1.76\frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts and $M = 16$, with $L = 1$ second, constant $\sum_{i=1}^{M} w_i = 5 \cdot 10^9 \frac{\text{cycles}}{\text{second}}$, and $w_1 = w_2 = \cdots = w_{M-1} = \frac{(\sum_{i=1}^{M} w_i) - w_M}{M-1}$.

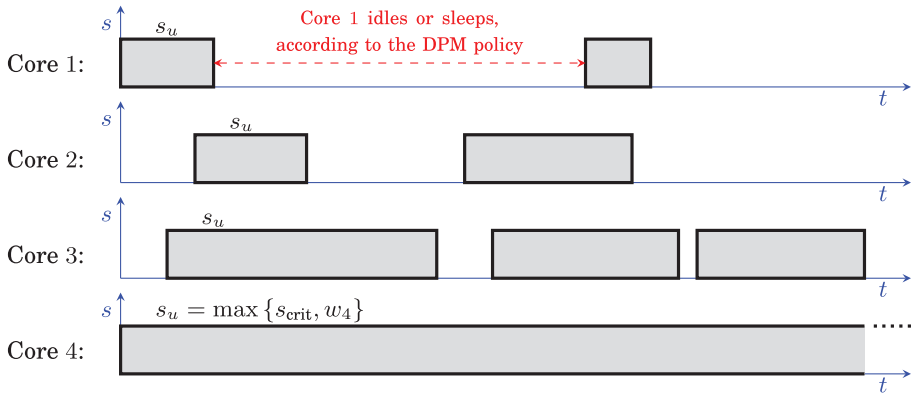


Fig. 5. Example for a voltage island with four cores using SFA. When a core has no workload to execute, it idles or sleeps according to the DPM policy (further details are presented in Section 8).

voltage island is

$$E_{\text{SFA}}(s_u) = L\left(\frac{\beta}{s_u} + \alpha s_u^{\gamma-1}\right)\sum_{i=1}^{M} w_i. \tag{10}$$

Function $E_{\text{SFA}}(s_u)$ is convex with respect to s_u and its first-order derivative with respect to s_u is the same as the first-order derivative of Eq. (2) with respect to s. Hence, the optimal s_u for SFA is also found at s_{crit}. Similarly to E^*, when $w_M \leq s_{\text{crit}}$, the aforesaid solution is a feasible one. Therefore, when $w_M \leq s_{\text{crit}}$, SFA is optimal and has the same energy consumption as the lower bound for the energy consumption E^*. For this reason, the relation between $s_{\min}$ and $\sqrt[\gamma]{\frac{\beta}{(\gamma-1)\alpha}}$ is of no consequence. Thus, for simplicity in presentation, we consider that $s_{\min} = 0$ and therefore set s_{crit} to $\sqrt[\gamma]{\frac{\beta}{(\gamma-1)\alpha}}$. This was implicit in Section 3 and will be considered until Section 9, inclusive. The approximation factors obtained for this condition are safe upper bounds for the general case.

Finally, the frequency chosen by SFA, namely s_u, is (1) s_{crit} if w_M is less than or equal to s_{crit}, and (2) w_M otherwise. In other words, $s_u = \max\{s_{\text{crit}}, w_M\}$. An example for a voltage island with four cores using SFA is presented in Figure 5.

By replacing s_u for the two cases in Eq. (10), we obtain the optimal energy consumption for SFA as a function of w_M, defined as $E_{\text{SFA}}(w_M)$ and presented

in Eq. (11).

$$E_{\text{SFA}}(w_M) = \begin{cases} \alpha\gamma L\left({s_{\text{crit}}}^{\gamma-1}\right)\sum_{i=1}^{M} w_i & \text{if } w_M \le s_{\text{crit}} \\ \frac{L}{w_M}\left(\beta + \alpha w_M^{\gamma}\right)\sum_{i=1}^{M} w_i, & \text{otherwise.} \end{cases} \tag{11}$$

In the case that $\beta = 0$, then s_{crit} is also zero and only the dynamic energy consumption is present for SFA, which is

$$E_{\text{SFA}}^{\beta=0}(w_M) = \alpha L\left({w_M}^{\gamma-1}\right)\sum_{i=1}^{M} w_i. \tag{12}$$

5. APPROXIMATION FACTOR FOR SFA WITH $\beta = 0$

This section presents the approximation factor of SFA when $\beta = 0$ defined as $\text{AF}_{\text{SFA}}^{\beta=0}$, using Eq. (5) and the dynamic energy consumption of SFA in Eq. (12). Note that, even though this is not a practical setting, we do incremental analysis for simplicity in presentation. The properties derived in this section will also be used in Section 6 when $\beta \neq 0$.

Since $0 = w_0 \le w_1 \le \cdots \le w_M$ from the problem definition, by introducing the scaling factor r_i we can rephrase the utilization w_i for $i = 0, 1, \ldots, M$ as $w_i = r_i \cdot w_M$, where

$$0 = r_0 < r_1 \le r_2 \le \cdots \le r_{M-1} \le r_M = 1. \tag{13}$$

The approximation factor of SFA when $\beta = 0$ can be expressed as

$$\text{AF}_{\text{SFA}}^{\beta=0} = \max H(r_0, \ldots, r_M), \tag{14}$$

where, for notational brevity,

$$H(r_0, \ldots, r_M) = \frac{\sum_{i=1}^{M} r_i}{\left[\sum_{i=1}^{M}(r_i - r_{i-1})\sqrt[\gamma]{M-i+1}\right]^{\gamma}}. \tag{15}$$

Intuitively, from Eqs. (11) and (12), when w_M and $\sum_{i=1}^{M} w_i$ are constant, SFA has a fixed energy consumption no matter how the cycle utilizations, namely w_1, $w_2, \ldots, w_{M-1}$, are distributed. However, for the lower bound of energy consumption, the utilization distribution matters. Specifically, we find a critical utilization distribution when $w_1 = w_2 = \cdots = w_{M-1} = \frac{\sum_{i=1}^{M-1} w_i}{M-1}$, resulting in a lower bound for Eqs. (5) and (9), as well as an upper bound for $H(r_0, \ldots, r_M)$. This is formally presented Lemma 5.1.

LEMMA 5.1. *For all $r_0, r_1 \ldots, r_M$ defined in Eq. (13) and, by defining δ as $\frac{\sum_{i=1}^{M-1} r_i}{M-1}$, with $\gamma > 1$, we have*

$$H(r_0, \ldots, r_M) \le h(\delta) = \frac{1 - \delta + \delta M}{\left(1 - \delta + \delta\sqrt[\gamma]{M}\right)^{\gamma}}. \tag{16}$$

PROOF. Suppose that we change the configuration from $r_1, \ldots, r_{M-1}$ to $r'_1, \ldots, r'_{M-1}$ such that $r_M = r'_M = 1$ and $\sum_{i=1}^{M-1} r_i = \sum_{i=1}^{M-1} r'_i$. Since $\sum_{i=1}^{M} r_i = \sum_{i=1}^{M} r'_i$, by Eq. (15), we know that $H(r'_0, \ldots, r'_M)$ under fixed $\sum_{i=1}^{M-1} r'_i$ is maximized if and only if $\sum_{i=1}^{M-1}(r'_i - r'_{i-1})\sqrt[\gamma]{M-i+1}$ under fixed $\sum_{i=1}^{M-1} r'_i$ is minimized. By using the extreme point theorem,

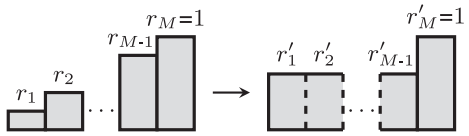


Fig. 6. Resulting cycle utilization relation change.

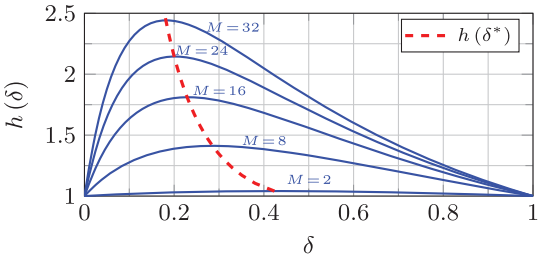


Fig. 7. $h(\delta)$ when $\gamma = 3$, highlighting δ^*.

it is not difficult to know that $\sum_{i=1}^{M-1}(r'_i - r'_{i-1})\sqrt[\gamma]{M-i+1}$ under a fixed $\sum_{i=1}^{M-1} r'_i$ has a global minimum when $\delta = \frac{\sum_{i=1}^{M-1} r'_i}{M-1} = r'_{i-1} = r'_i$ for all $r'_1, \ldots, r'_{M-1}$ (as shown in Figure 6). By setting r_i to δ for $i = 1, 2, \ldots, M-1$ in Eq. (15), the lemma is proven. □

By taking the first-order derivative of $h(\delta)$ with respect to δ, defined in Lemma 5.1, it can be easily seen that $h(\delta)$ is a convex function of δ when $\gamma > 1$ and its maximum value happens when δ is δ^*, defined as

$$\delta^* = \frac{\gamma - 1 + M - \gamma\sqrt[\gamma]{M}}{(\gamma-1)(M\sqrt[\gamma]{M} - M - \sqrt[\gamma]{M} + 1)}. \tag{17}$$

A representation of $h(\delta)$ when γ is 3 can be seen in Figure 7.

Finally, when w_M and $\sum_{i=1}^{M} w_i$ are fixed, the approximation factor of SFA is maximized when $w_1 = w_2 = \cdots = w_{M-1} = \delta^* \cdot w_M$. This is formally expressed in Theorem 5.2.

THEOREM 5.2. *When $\beta = 0$, the approximation factor $AF_{SFA}^{\beta=0}$ of SFA for periodic real-time tasks is*

$$AF_{SFA}^{\beta=0} \le h(\delta^*), \tag{18}$$

where δ^* is defined as a function of γ and M in Eq. (17) and where $h(\delta)$ is defined in Eq. (16). Since $h(\delta^*)$ only depends on γ and M, $AF_{SFA}^{\beta=0}$ is independent of the value of α.

PROOF. Based on the definition of function $h(\delta)$ in Lemma 5.1, the definition of $H(r_0, \ldots, r_M)$ in Eq. (15), and the relation between $\mathrm{AF}_{\mathrm{SFA}}^{\beta=0}$ and $H(r_0, \ldots, r_M)$ in Eq. (14), we can express $\mathrm{AF}_{\mathrm{SFA}}^{\beta=0}$ as a function of δ^* to obtain the inequality in Eq. (18). That is,

$$\mathrm{AF}_{\mathrm{SFA}}^{\beta=0} = \max H(r_0, \ldots, r_M) \le \max h(\delta) \le h\left(\delta^*\right),$$

and thus the theorem is proven. □

A representation of $\mathrm{AF}_{\mathrm{SFA}}^{\beta=0}$ for different values of M when $\gamma = 2$ and $\gamma = 3$ can be seen in Figure 8.

6. APPROXIMATION FACTOR FOR SFA WITH $\beta \neq 0$

After deriving the lower bound of energy consumption that considers the leakage in Eq. (9) and the energy consumption of SFA in Eq. (11), this section presents the analysis for the approximation factor, defined in Eq. (3), for SFA with $\beta \neq 0$. We will first present the analysis based on a given s_{dyn} as defined in Lemma 3.1. Then, we will analyse the approximation factor based on the critical utilization distribution among the task sets to provide a safe factor.

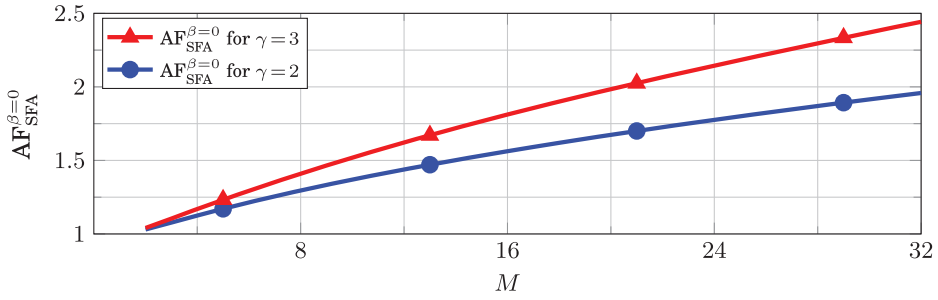


Fig. 8. Approximation factor for SFA with $\beta = 0$.

6.1. Approximation Factor as a Function of s_{dyn}

Replacing inside Eq. (3) the different possible values of E_{SFA} from Eq. (11) and of E^* from Eq. (9) in Lemma 3.1, we get the relation $\frac{E_{\text{SFA}}}{E^*}$ as a function of w_M as

$$\frac{E_{\text{SFA}}}{E^*}(w_M) = \begin{cases} \frac{\beta+\alpha w_M^{\gamma}}{w_M \alpha \gamma (s_{\text{crit}}{}^{\gamma-1})} & \text{if } s_{\text{crit}} < w_M \leq s_{\text{dyn}} \\ \frac{\frac{\beta+\alpha w_M^{\gamma}}{\alpha w_M} \sum_{i=1}^{M} w_i}{\left[\sum_{i=1}^{M}(w_i - w_{i-1}) \sqrt[\gamma]{M-i+1}\right]^{\gamma}} & \text{if } s_{\text{dyn}} < w_M < s_{\text{max}} \\ 1 & \text{otherwise.} \end{cases} \tag{19}$$

By using the scaling factor r_i from Eq. (13) and the definition of $H(r_0, \ldots, r_M)$ from Eq. (15), we have the following lemma for the approximation factor.

LEMMA 6.1. *By defining* s_{dyn} *as*

$$s_{dyn} = s_{crit}[\gamma H(r_0, \ldots, r_M)]^{\frac{1}{\gamma-1}}, \tag{20}$$

we have

$$\frac{E_{SFA}}{E^*}(w_M) \leq \frac{1}{\alpha\gamma(s_{crit}{}^{\gamma-1})}\left(\frac{\beta}{s_{dyn}} + \alpha {s_{dyn}}^{\gamma-1}\right). \tag{21}$$

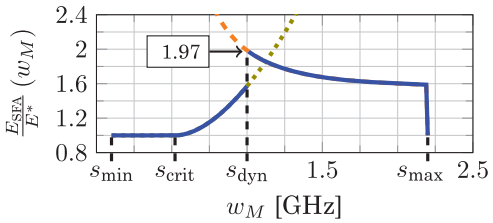
PROOF. Since α, β, γ and s_{crit} are all constants, we know that $\frac{E_{\text{SFA}}}{E^*}(w_M)$ is a convex and increasing function with respect to w_M when $s_{\text{crit}} < w_M \leq s_{\text{dyn}}$. Therefore, for this case, $\frac{E_{\text{SFA}}}{E^*}(w_M)$ is less than or equal to $\frac{E_{\text{SFA}}}{E^*}(s_{\text{dyn}})$.

With r_i from Eq. (13) and the definition of $H(r_0, \ldots, r_M)$ from Eq. (15) we can rephrase Eq. (19) when $s_{\text{dyn}} < w_M < s_{\text{max}}$ to

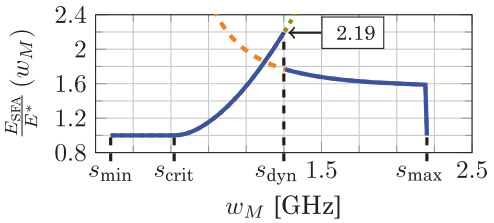
$$\frac{\frac{\beta}{w_M^{\gamma}} + \alpha}{\alpha} H(r_0, \ldots, r_M). \tag{22}$$

Clearly, for a given task partitioning, the cycle utilization relations between task sets are fixed, therefore, $H(r_0, \ldots, r_M)$ is also constant for a given task partitioning. Together with the fact that α, β, M, and γ are constants, we know that Eq. (22) is a decreasing function with respect to w_M.

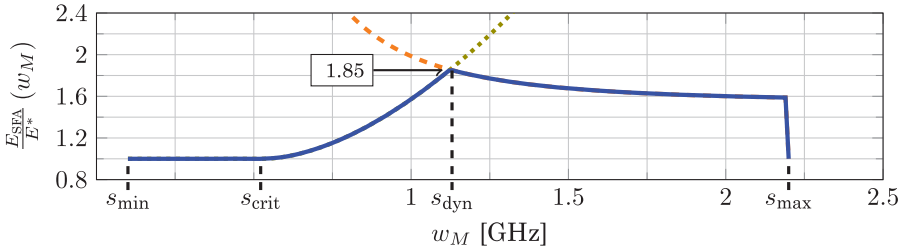
Examples for the approximation factor function $\frac{E_{\text{SFA}}}{E^*}(w_M)$, for different choices of s_{dyn}, can be seen in Figure 9. Particularly, the examples in Figure 9(a) and Figure 9(b) show bad choices for s_{dyn} that would result in pessimistic approximation factors. With the preceding analysis, function $\frac{E_{\text{SFA}}}{E^*}(w_M)$ reaches the lowest upper bound when the



(a) s_{dyn} that would result in a pessimistic approximation factor



(b) s_{dyn} that would result in a pessimistic approximation factor



(c) lowest upper bound by properly choosing s_{dyn}

Fig. 9. $\frac{E_{SFA}}{E^*}(w_M)$ when $\alpha = 1.76\frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts, and $M = 16$ with $L = 1$ second, as well as $w_i = 0.51 \cdot w_M$ for all $i = 1, \ldots, M-1$, considering different choices for s_{dyn}.

aforesaid two cases intersect each other, as shown in the example in Figure 9(c), that is, when

$$\alpha\gamma\left({s_{crit}}^{\gamma-1}\right)w_M\sum_{i=1}^{M}r_i = \alpha w_M^{\gamma}\left[\sum_{i=1}^{M}(r_i - r_{i-1})\sqrt[\gamma]{M-i+1}\right]^{\gamma},$$

resulting in the definition of s_{dyn} in Eq. (20). Thus, Eq. (21) holds and this lemma is proven. □

6.2. Approximation Factor for the Critical Utilization Distribution

Lemma 6.1 presents the analysis of relation $\frac{E_{SFA}}{E^*}(w_M)$ for a given cycle utilization distribution, namely $w_1, w_2, \ldots, w_M$. To obtain the approximation factor from Eq. (3), we rephrase this relation for the critical utilization distribution (also used when $\beta = 0$) presented in Lemma 5.1.

From Lemma 5.1, we rewrite Eq. (20) as a function of $h(\delta^*)$ and define s^*_{dyn} as the upper bound for the value of s_{dyn} as

$$s^*_{dyn} = s_{crit}[\gamma h(\delta^*)]^{\frac{1}{\gamma-1}}. \tag{23}$$

The following theorem concludes the analysis of AF_{SFA}.

THEOREM 6.2. *When $\beta \neq 0$, the approximation factor AF_{SFA} for periodic real-time tasks is*

$$AF_{SFA} \leq \frac{\gamma - 1}{[\gamma^{\gamma} h(\delta^*)]^{\frac{1}{\gamma-1}}} + h(\delta^*), \tag{24}$$

where δ^* is defined as a function of γ and M in Eq. (17) where and $h(\delta)$ is defined in Eq. (16). Since $h(\delta^*)$ is only a function of γ and M, AF_{SFA} is independent of α and β.

PROOF. From the definition of AF_{SFA} in Eq. (3) based on Eq. (21) in Lemma 6.1, Eq. (16) in Lemma 5.1, and Eq. (23) as well as the definitions ${s_{\text{crit}}}^{\gamma} = \frac{\beta}{(\gamma-1)\alpha}$ and function $h(\delta)$, we can replace s^*_{dyn} as a function of δ^* to obtain the inequality in Eq. (24). In other words,

$$\begin{aligned}
\text{AF}_{\text{SFA}} \leq \frac{\beta + \alpha {s_{\text{dyn}}}^{\gamma}}{s_{\text{dyn}}\alpha\gamma\left({s_{\text{crit}}}^{\gamma-1}\right)} &= \frac{\beta + \alpha {s_{\text{crit}}}^{\gamma}\left\{[\gamma h(\delta^*)]^{\frac{1}{\gamma-1}}\right\}^{\gamma}}{s_{\text{crit}}[\gamma h(\delta^*)]^{\frac{1}{\gamma-1}}\alpha\gamma\left({s_{\text{crit}}}^{\gamma-1}\right)} \\
&= \frac{\alpha(\gamma-1)\frac{\beta}{(\gamma-1)\alpha} + \alpha {s_{\text{crit}}}^{\gamma}\left\{[\gamma h(\delta^*)]^{\frac{1}{\gamma-1}}\right\}^{\gamma}}{[\gamma h(\delta^*)]^{\frac{1}{\gamma-1}}\alpha\gamma\left({s_{\text{crit}}}^{\gamma}\right)} \\
&= \frac{(\gamma-1) + \left\{[\gamma h(\delta^*)]^{\frac{1}{\gamma-1}}\right\}^{\gamma}}{[\gamma h(\delta^*)]^{\frac{1}{\gamma-1}}\gamma} = \frac{\gamma-1}{[\gamma^{\gamma} h(\delta^*)]^{\frac{1}{\gamma-1}}} + h(\delta^*),
\end{aligned}$$

and thus the theorem is proven. □

7. APPROXIMATION FACTOR FOR SFA FOR BALANCED TASK SETS

This section analyses the approximation factor of SFA when the system uses a load balancer for task partitioning.

LEMMA 7.1. *Given* $0 < r_1 \leq r_2 \leq \cdots \leq r_M = 1$, $M \geq 2$ *and* $\gamma > 1$, *if* $\frac{\sum_{i=1}^{M-1} r_i}{M-1} \geq 0.5$, *then* $h(\delta) \leq h(0.5)$.

PROOF. From Eq. (16) (illustrated in Figure 7), it is clear that $h(\delta)$ is a decreasing function with respect to δ when $\delta^* \leq \delta \leq 1$, $M \geq 2$, and $\gamma > 1$. By taking the first-order derivative of δ^* with respect to M from Eq. (17), it can be easily proven that δ^* is a decreasing function with respect to M, since $\frac{\partial \delta^*}{\partial M} \leq 0$ for all $M \geq 2$ and $\gamma > 1$. Hence, the highest value of δ^* for a given γ occurs when $M = 2$, which we define as $\delta^*_{M=2} = \frac{\gamma - \gamma\sqrt[\gamma]{2}+1}{(\gamma-1)(\sqrt[\gamma]{2}-1)}$.

Furthermore, by taking the first-order derivative of δ^* with respect to γ from Eq. (17), it can also be proven that δ^* is an increasing function with respect to γ, since $\frac{\partial \delta^*}{\partial \gamma} \geq 0$ for all $M \geq 2$ and $\gamma > 1$. Therefore, the highest value of δ^* is obtained when $M = 2$, that is, for $\delta^*_{M=2}$, and $\gamma \to \infty$, which converges to $\frac{1}{\ln 2} - 1 = 0.443$. Finally, since $\delta^* < 0.5$ for any $M \geq 2$ and $\gamma > 1$, then $h(\delta)$ is a decreasing function after 0.5 for any $M \geq 2$ and $\gamma > 1$, thus the lemma is proven. □

THEOREM 7.2. *Given* $M \geq 2$, *if* δ, *defined in Lemma 5.1 as* $\frac{\sum_{i=1}^{M-1} r_i}{(M-1)}$, *is no less than* 0.5, *then the approximation factor of SFA for periodic real-time tasks is*

$$AF^{\beta=0}_{SFA}(\delta \geq 0.5) \leq h(0.5), \tag{25}$$

or

$$AF_{SFA}(\delta \geq 0.5) \leq \frac{\gamma-1}{\left[\gamma^{\gamma} h(0.5)\right]^{\frac{1}{\gamma-1}}} + h(0.5), \tag{26}$$

when $\beta = 0$ *and* $\beta \neq 0$, *respectively.*

PROOF. This is based on Lemma 7.1, Theorem 5.2, and Theorem 6.2. □

COROLLARY 7.3. *Clearly,* δ *plays a major role in the approximation factor of SFA. Even though we do not analyse task partitioning for SFA, from Theorem 7.2 we can*

conclude that, if the system uses a load balancer for task partitioning, this would lead to a better approximation factor for SFA. Technically, if the number of execution cycles of any task is no more than that of the average execution cycles on all cores, a load balancer like the largest-task-first strategy in Yang et al. [2005] results in $\frac{w_1}{w_M} \geq 0.5$ *(as proved in Yang et al. [2005, Lemma 5]).*

8. APPROXIMATION FACTOR FOR SFA WITH NON-NEGLIGIBLE SWITCHING OVERHEAD

When the energy overhead for entering/leaving a low-power mode is nonnegligible, we cannot always switch a core to a low-power mode immediately when there is no workload on it to execute. We have to consider the *break-even time*, defined as the time such that the energy consumption for idling (in execution mode) is the same as the overhead for entering and leaving a low-power mode.

The SFA strategy can be combined with any unicore procrastination algorithm in the literature to decide when to switch a core to a low-power mode, such as in Irani et al. [2003] and Chen and Kuo [2007], and the analysis for combining SFA with each algorithm should accordingly be studied.

8.1. Approximation Factor of Using SFA Combined with Algorithm Left-to-Right (LTR)

Particularly, this section analyses the approximation factor of using SFA combined with algorithm Left-to-Right (LTR) from Irani et al. [2003] against the optimal DVFS and DPM solution. That is, we use SFA to decide the voltage/frequency of the island and use LTR to decide whether/when each individual core should sleep/activate.

For notational brevity, we isolate certain portions of energy consumption for a schedule S, as done in Irani et al. [2003].

—*energy* (S). This is the total energy consumed by schedule S during a hyperperiod.
—*active* (S). This is the energy expended while the system is active, that is, the energy consumption for *executing tasks*.
—*idle* (S). It is the cost to keep the system active or enter and leave a low-power mode during idle periods (depending on which action is the most energy efficient).
—*on* (S). This is the cost to keep the system in the *on* state while the system is on.
—*sleep* (S). It is the cost to leave the low-power mode at the end of each sleep interval.

For the rest of this section, we define S_{OPT} as an optimal DVFS and DPM schedule, whereas $S_{\text{OPT},m}$ is the corresponding schedule by considering only the m-th core. Similarly, we define $S^{\text{SFA}}_{\text{LTR}}$ as the schedule made using SFA for executing and LTR for sleeping, whereas $S^{\text{SFA}}_{\text{LTR},m}$ is the corresponding schedule made by considering only the m-th core.

According to Theorem 6.2, we know that

$$\text{active}\left(S^{\text{SFA}}_{\text{LTR}}\right) \leq \text{AF}_{\text{SFA}} \cdot \text{active}\left(S_{\text{OPT}}\right). \tag{27}$$

Additionally, independently from the task execution on a single core m, adopting LTR on the core ensures that $\text{idle}(S^{\text{SFA}}_{\text{LTR},m}) \leq \text{on}(S_{\text{OPT},m}) + 2 \cdot \text{sleep}(S_{\text{OPT},m})$ (i.e., from Irani et al. [2003, Lemma 10]). By using the summation of all the cores in the island, we have

$$\text{idle}\left(S^{\text{SFA}}_{\text{LTR}}\right) \leq \text{on}\left(S_{\text{OPT}}\right) + 2 \cdot \text{sleep}\left(S_{\text{OPT}}\right). \tag{28}$$

THEOREM 8.1. *When* $P(s)$ *is a convex and increasing function, combining SFA and LTR results in an approximation factor, against the optimal DVFS and DPM solution, defined as* $AF_{SFA\text{-}LTR}$*, less than or equal to* $AF_{SFA} + 1$.

PROOF. From Eqs. (27) and (28) and considering that by definition $\mathrm{AF}_{\mathrm{SFA}} \geq 1$, we have

$$\begin{aligned}
\mathrm{energy}\big(S^{\mathrm{SFA}}_{\mathrm{LTR}}\big) &= \mathrm{active}\big(S^{\mathrm{SFA}}_{\mathrm{LTR}}\big) + \mathrm{idle}\big(S^{\mathrm{SFA}}_{\mathrm{LTR}}\big) \\
&\leq \mathrm{AF}_{\mathrm{SFA}} \cdot \mathrm{active}\,(S_{\mathrm{OPT}}) + \mathrm{on}\,(S_{\mathrm{OPT}}) + 2 \cdot \mathrm{sleep}\,(S_{\mathrm{OPT}}) \\
&\leq \mathrm{AF}_{\mathrm{SFA}} \cdot \mathrm{active}\,(S_{\mathrm{OPT}}) + \mathrm{active}\,(S_{\mathrm{OPT}}) + \mathrm{idle}\,(S_{\mathrm{OPT}}) + \mathrm{sleep}\,(S_{\mathrm{OPT}}) \\
&\leq (\mathrm{AF}_{\mathrm{SFA}} + 1) \cdot \mathrm{active}\,(S_{\mathrm{OPT}}) + 2 \cdot \mathrm{idle}\,(S_{\mathrm{OPT}}) \\
&\leq \max\{\mathrm{AF}_{\mathrm{SFA}} + 1, 2\} \cdot \mathrm{energy}\,(S_{\mathrm{OPT}}) \\
&\leq (\mathrm{AF}_{\mathrm{SFA}} + 1) \cdot \mathrm{energy}\,(S_{\mathrm{OPT}})\,.
\end{aligned}$$

Hence, the theorem is proven. □

8.2. Approximation Factor of Using SFA Combined with the Optimal DPM Solution

In Theorem 8.2, this section extends the results in Section 8.1 in order to analyse the approximation factor of using SFA combined with the optimal DPM solution (for such a case) [Baptiste et al. 2012] against the optimal DVFS and DPM solution.

We define $S^{\mathrm{SFA}}_{\mathrm{DPM}^*}$ as the schedule obtained by using SFA for executing and the optimal DPM solution for sleeping, whereas $S^{\mathrm{SFA}}_{\mathrm{DPM}^*,m}$ is the corresponding schedule by considering only the m-th core. Furthermore, Eq. (28) can be extended to

$$\mathrm{idle}\big(S^{\mathrm{SFA}}_{\mathrm{DPM}^*}\big) \leq \mathrm{idle}\big(S^{\mathrm{SFA}}_{\mathrm{LTR}}\big) \leq \mathrm{on}\,(S_{\mathrm{OPT}}) + 2 \cdot \mathrm{sleep}\,(S_{\mathrm{OPT}})\,. \tag{29}$$

THEOREM 8.2. *When $P(s)$ is a convex and increasing function, combining SFA with the optimal DPM solution (for such a case) results in an approximation factor, against the optimal DVFS and DPM solution, defined as $AF_{SFA\text{-}DPM^*}$, less than or equal to $AF_{SFA} + 1$.*

PROOF. From Eq. (27) (for $S^{\mathrm{SFA}}_{\mathrm{DPM}^*}$ instead of $S^{\mathrm{SFA}}_{\mathrm{LTR}}$) and Eq. (29), the proof of this theorem is similar to that of Theorem 8.1. □

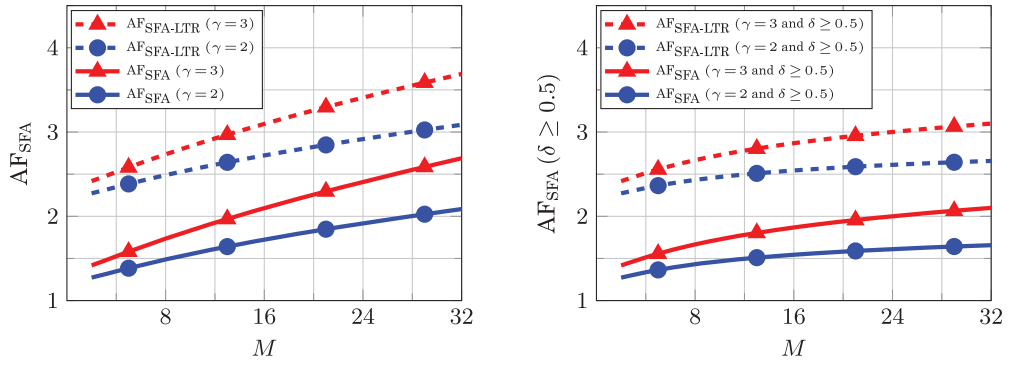
9. NUMERICAL RESULTS FOR WORST CASES

This section presents numerical results for the approximation factor of SFA for the *critical utilization distribution*.

As stated in Theorem 5.2 and Theorem 6.2, the approximation factor of SFA depends on γ and M. Hence, we consider some practical settings for γ, namely $\gamma = 2$ and $\gamma = 3$, and explore the impact of M on the approximation factor. Theoretically, the approximation factor can go up to ∞ when $M \to \infty$. However, practically speaking, the number of cores in a voltage island is not a very large number, thus we would like to explore the applicability of SFA for a limited number of cores per island.

Figure 10(a) presents the approximation factor, based on Theorem 6.2, for M up to 32 when $\gamma = 2$ and $\gamma = 3$. Whenever the voltage island has at most 4 (8, 16, 32, respectively) cores and negligible overhead for sleeping is considered, SFA has an approximation factor that can be bounded to at most 1.53 (1.74, 2.10, 2.69, respectively) when $\gamma = 3$ and to at most 1.35 (1.49, 1.73, 2.09, respectively) when $\gamma = 2$. When we consider nonnegligible overhead for sleeping, from Theorem 8.1, these values are incremented by 1.

Motivated by the conclusions from Corollary 7.3, Figure 10(b) shows the approximation factor for SFA under the condition $\frac{w_1}{w_M} \geq 0.5$ based on Theorem 7.2, that is, $\delta = 0.5$. Practically, this leads to a better approximation factor for SFA. When $\frac{w_1}{w_M} \geq 0.5$ and the voltage island has at most 4 (8, 16, 32, respectively) cores and negligible overhead for sleeping is considered, SFA has an approximation factor that can be bounded to at most 1.52 (1.67, 1.87, 2.10, respectively) when $\gamma = 3$ and to at most 1.34 (1.44, 1.55,



(a) worst case (b) with $\delta \geq 0.5$

Fig. 10. Approximation factors for SFA.

1.66, respectively) when $\gamma = 2$. When we consider nonnegligible overhead for sleeping, from Theorem 8.1, these values are incremented by 1.

10. SYSTEMS WITH DISCRETE FREQUENCIES

This section presents a simple extension for systems with power consumption function $P(s) = \beta + \alpha s^\gamma$, from Eq. (1), and with discrete frequencies $\{f_1, f_2, \ldots, f_F\}$ such that $f_1 = s_{\min}$ and $f_F = s_{\max}$. For notational brevity, we define the auxiliary frequencies $f_0 = 0$ and $f_{F+1} \to \infty$.

For such systems, given the convexity of $\frac{P(s)}{s}$, the lower bound of the energy consumption by considering continuous frequencies, namely E^*, is also a lower bound for the optimal energy consumption under discrete frequencies.

To meet the timing constraints, the discrete execution frequency for SFA, denoted as f_i, will simply be chosen among all the available frequencies such that f_i is the lowest frequency $\in \{f_1, f_2, \ldots, f_F\}$ with $f_i \geq s_u = \max\{s_{\text{crit}}, w_M\}$. Moreover, the following lemma relates the ratio between the energy consumption of SFA for discrete frequencies, denoted as $E_{\text{SFA}}^{\text{discrete}}$, and the energy consumption of SFA for continuous frequencies, previously defined as E_{SFA} in Eq. (11).

LEMMA 10.1. *When $f_{i-1} < s_u \leq f_i$, the energy consumption of SFA for execution by running at frequency f_i is equal to the energy consumption of SFA for execution by running at frequency s_u, multiplied with $\theta(s_u) = \frac{P(f_i)\cdot s_u}{P(s_u)\cdot f_i}$.*

PROOF. From Eq. (11), $E_{\text{SFA}} = L\frac{P(s_u)}{s_u}\sum_{i=1}^{M} w_i$. Similarly, when considering discrete frequencies, $E_{\text{SFA}}^{\text{discrete}} = L\frac{P(f_i)}{f_i}\sum_{i=1}^{M} w_i$ such that $f_{i-1} < s_u \leq f_i$. Therefore, $\frac{E_{\text{SFA}}^{\text{discrete}}}{E_{\text{SFA}}} = \frac{P(f_i)\cdot s_u}{P(s_u)\cdot f_i} = \theta(s_u)$ and the lemma is proven. □

From Lemma 10.1 we can derive the following theorem for the approximation factor of SFA for systems with discrete frequencies.

THEOREM 10.2. *For systems with power consumption function $P(s) = \beta + \alpha s^\gamma$ and with discrete available frequencies $\{f_1, f_2, \ldots, f_F\}$ such that $f_1 = s_{min}$ and $f_F = s_{max}$, the approximation factor of SFA for discrete frequencies, defined as $AF_{SFA}^{discrete}$, is*

$$AF_{SFA}^{discrete} \leq AF_{SFA} \cdot \theta_{max}$$

with

$$\theta_{max} = \max\left\{\frac{P(f_h)\cdot s_{crit}}{P(s_{crit})\cdot f_h}, \max_{h<i\leq F}\frac{P(f_i)\cdot f_{i-1}}{P(f_{i-1})\cdot f_i}\right\} \qquad \textit{such that } f_{h-1} < s_{crit} \leq f_h,$$

where f_h is the lowest available discrete frequency higher than s_{crit} and where θ_{max} depends on the hardware parameters α, γ and β, and also on the set of available frequencies.

PROOF. For a given w_M, the approximation factor of SFA for discrete frequencies can be expressed as

$$\mathrm{AF}_{\mathrm{SFA}}^{\mathrm{discrete}} \leq \max\frac{E_{\mathrm{SFA}}^{\mathrm{discrete}}}{E^*} = \max\left\{\frac{E_{\mathrm{SFA}}}{E^*}\cdot\frac{E_{\mathrm{SFA}}^{\mathrm{discrete}}}{E_{\mathrm{SFA}}}\right\} = \mathrm{AF}_{\mathrm{SFA}}\cdot\max\{\theta(s_u)\}$$

for which the worst case of $\theta(s_u)$ for any possible s_u, is denoted as $\theta_{\max} = \max\{\theta(s_u)\}$.

Let frequency f_h be the lowest available discrete frequency higher than s_{crit}, that is, $f_{h-1} < s_{\mathrm{crit}} \leq f_h$. Because $\frac{P(s)}{s}$ is a convex and nondecreasing function when $s \geq s_{\mathrm{crit}}$, we know that, for a given s_u, the highest value of $\theta(s_u)$ happens: (1) when the value of s_u is close to f_{i-1}, if $f_h \leq f_{i-1}$; or (2) when $s_u = s_{\mathrm{crit}}$, otherwise. Therefore, $\theta_{\max}$ is the maximum value among all such cases and the theorem is proven. □

For example, based on Theorem 10.2, for a system with $\alpha = 1.76\frac{\mathrm{Watts}}{\mathrm{GHz}^3}$, $\beta = 0.5$ Watts, $\gamma = 3$, $s_{\mathrm{crit}} = 0.52$ GHz, and available frequencies $\{0.1\text{ GHz}, 0.2\text{ GHz}, \ldots, 3.0\text{ GHz}\}$, the value of $\theta_{\max}$ is about 1.14.

11. SYSTEMS WITH MULTIPLE VOLTAGE ISLANDS

The analysis of SFA in a single voltage island, in terms of energy consumption, can be easily extended to consider multicore systems with multiple voltage islands as stated in the following theorem.

THEOREM 11.1. *For a system with V voltage islands under a given mapping of task partitions, that is, each island has a set of task sets already assigned to it and these task sets cannot be swapped between islands, the approximation factor (with respect to the energy consumption) by running each voltage island with SFA against running each island with the optimal DVFS schedule, denoted as $AF_{SFA}^{V\text{-}islands}$, is equal to the approximation factor of SFA in an individual voltage island, namely AF_{SFA}.*

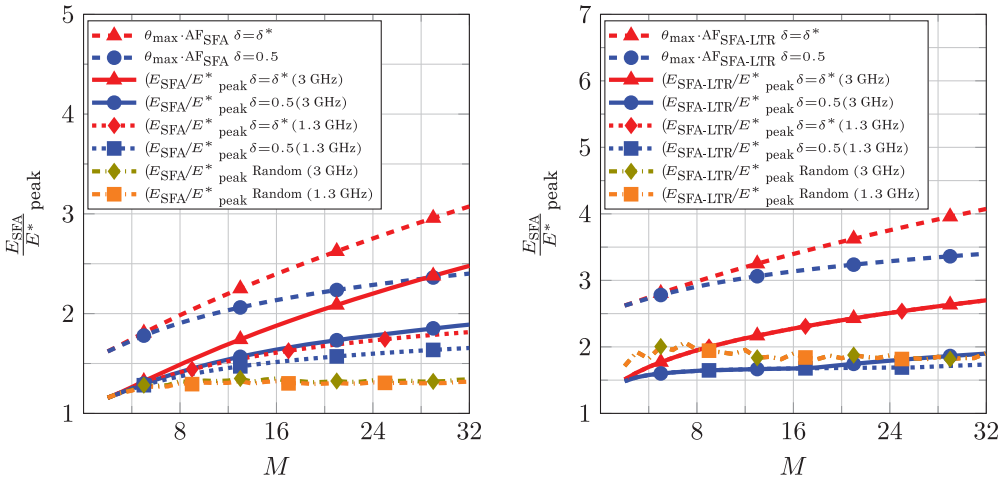
PROOF. The proof comes directly from the definition of $\mathrm{AF}_{\mathrm{SFA}}$ in Eq. (3), that is,

$$\mathrm{AF}_{\mathrm{SFA}}^{\text{V-islands}} = \max\frac{\sum_{j=1}^{V} E_{\mathrm{SFA}\,j}}{\sum_{j=1}^{V} E_{\mathrm{OPT}\,j}} \leq \frac{\sum_{j=1}^{V}\mathrm{AF}_{\mathrm{SFA}}\cdot E_j^*}{\sum_{j=1}^{V} E_j^*} = \mathrm{AF}_{\mathrm{SFA}},$$

where $E_{\mathrm{SFA}\,j}$, $E_{\mathrm{OPT}\,j}$, and E_j^* are the energy consumptions for SFA, for an optimal schedule, and for the lower bound, all three during a hyperperiod on voltage island j. □

12. SIMULATIONS

This section simulates the performance of SFA for different scenarios in a single voltage island. Instead of analysing the approximation factor by using Theorem 6.2, we use Newton's method for solving Eq. (8) for a given input instance, thus providing a *concrete factor* based on the energy consumption of SFA.



(a) negligible overhead for sleeping

(b) nonnegligible overhead for sleeping

Fig. 11. Simulation results with $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts, $\gamma = 3$.

12.1. Simulation Setup

The parameters of $P(s)$ are chosen as $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts, and $\gamma = 3$, resulting in $s_{\text{crit}} = 0.52$ GHz, modelled from the experimental measurements from Howard et al. [2011] as explained in Section 2.1. We consider three cases for the cycle utilization distributions of the task sets: (1) the theoretical critical utilization distribution from Lemma 5.1 with $\delta = \delta^*$, (2) the critical utilization distribution based on Lemma 5.1 with $\delta = 0.5$ (for balanced task partitions), and also (3) 100 different random utilization distributions for every M. For all three utilization distributions, we consider w_M stepped by 20 MHz in the range of: (a) $[0.2 \text{ GHz}; 1.3 \text{ GHz}]$ and (b) $[0.2 \text{ GHz}; 3.0 \text{ GHz}]$, with different hyperperiods $L = 1, 2, \ldots, 5$ seconds for every w_M. Case (a) corresponds to the practical setting of (noncontinuous) available frequencies $\{0.1 \text{ GHz}, 0.2 \text{ GHz}, \ldots, 1.3 \text{ GHz}\}$ in the 48-core system from Howard et al. [2011]; higher utilization values would lead to infeasible solutions for partitioned scheduling using such platform. However, we would like to test SFA for higher utilizations (particularly not so close to s_{crit}), which is the reason why we also consider case (b), which is a hypothetical platform with the same power parameters but with available frequencies $\{0.1 \text{ GHz}, 0.2 \text{ GHz}, \ldots, 3.0 \text{ GHz}\}$.

For distributions (1) and (2), the maximum concrete factor among these (a) 280 and (b) 705 settings for w_M and L is reported as the peak factor for approximation, denoted by $\frac{E_{\mathrm{SFA}}}{E^*}_{\mathrm{peak}}$. For random distributions (3), the peak factor is taken from the: (a) $2.8 \cdot 10^4$ and (b) $7.05 \cdot 10^4$ values for every M.

12.2. Simulation Results

Figure 11(a) presents the results of $\frac{E_{\mathrm{SFA}}}{E^*}_{\mathrm{peak}}$ for the six configurations, namely (1a), (1b), (2a), (2b), (3a), and (3b), together with the analytical upper bounds $\theta_{\max} \cdot \mathrm{AF}_{\mathrm{SFA}}$ derived from Theorem 6.2 and Theorem 7.2 for $\gamma = 3$, with $\theta_{\max} = 1.14$. Cases (1b) and (2b) provide a lower bound for $\mathrm{AF}_{\mathrm{SFA}}$ when $\delta = \delta^*$ and $\delta = 0.5$, respectively. The difference between these values and the theoretical values of $\theta_{\max} \cdot \mathrm{AF}_{\mathrm{SFA}}$ is at most 0.62 for each case. This means that all the pessimism introduced in our analysis to obtain a safe upper bound for the approximation factor of SFA does not provide results so far away from simulated concrete cases. In fact, the theoretical $\theta_{\max} \cdot \mathrm{AF}_{\mathrm{SFA}}$ is not as

pessimistic as the assumptions lead to believe, even for discrete frequencies. For cases (1a) and (2a), the difference against the theoretical factor is much larger, resulting not only because of the precise computation of E^* by using Newton's method to solve Eq. (8), but because $s_{\max}$ is close to s_{crit} (2.5 times its value). This result, although making our analysis pessimistic for such a practical consideration, further supports the effectiveness of SFA for such platforms, such as SCC [Intel 2009]. Moreover, for cases (3a) and (3b), $\frac{E_{\text{SFA}}}{E^*}_{\text{peak}}$ is no more than 1.5, suggesting that for general cases SFA is a simple and effective scheme for energy minimization and that complicated DVFS solutions are not necessary unless better results for the worst case are required.

Figure 11(b) presents similar results as Figure 11(a), but for frame-based tasks using SFA in combination with LTR. The nonnegligible energy overhead for entering/leaving a low-power mode is set to 0.2 Joule. However, for computing E^* we only consider the energy consumption for executing, by ignoring the overhead for entering/leaving a low-power mode that is a lower bound for the optimal energy consumption. The reason why we choose this lower bound is due to the difficulty in deriving tighter lower bounds when $\beta \neq 0$ and when nonnegligible energy overhead for entering/leaving a low-power mode is considered. Especially such ignorance in our setting also leads to the significantly higher factor $\frac{E_{\text{SFA-LTR}}}{E^*}_{\text{peak}}$ for cases (3a) and (3b) when $M \leq 6$ in Figure 11(b). This happens because, for such cases, the energy consumption for executing results in low values both for SFA and the lower bound, which are shadowed by only considering the overhead for entering/leaving a low-power mode for SFA. Moreover, the effect of the overhead is more important than the relation of s_{crit} and $s_{\max}$, since similar results are obtained for w_M settings (a) and (b) for all three distributions (1), (2), and (3).

13. APPROXIMATION FACTOR FOR SFA CONSIDERING TASK PARTITIONING

The work in Pagani and Chen [2013] uses the results presented in this article and extends our theoretical analysis to derive the approximation factor of SFA combined with a task partitioning strategy, called the *Double-Largest-Task-First* (DLTF) strategy, against the optimal task partitioning and optimal DVFS schedule in terms of energy efficiency. For completeness, this section summarizes the results from Pagani and Chen [2013].

The *Double-Largest-Task-First* (DLTF) strategy is based on the *Largest-Task-First* strategy, which in turn is mainly a reformulation of the *Longest-Processing-Time* (LPT) algorithm from Graham [1969] for the makespan problem. The approximation factor of LTF in terms of task partitioning, denoted as ψ_{LTF}, is expressed as

$$\psi_{\text{LTF}} = \frac{4}{3} - \frac{1}{3M}$$

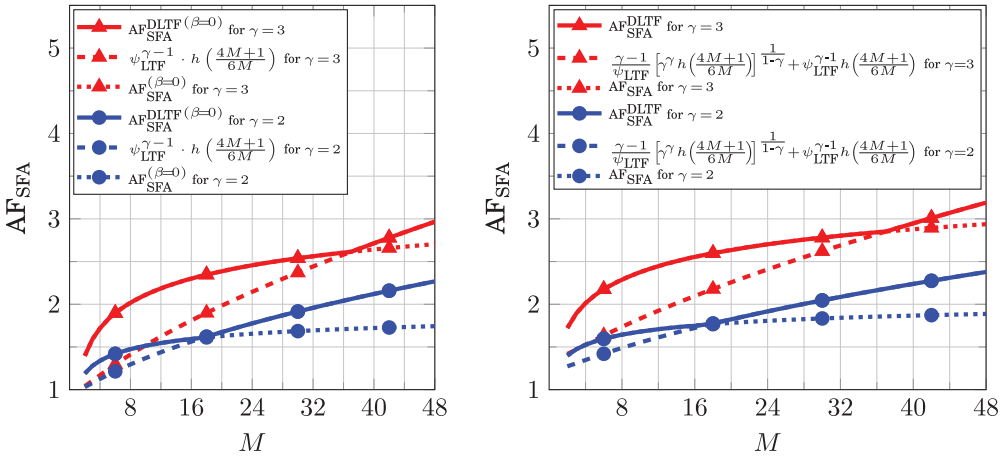
due to the approximation factor of LPT for the makespan problem from Graham [1969].

With this consideration, the approximation factor for DLTF combined with SFA in terms of energy consumption, when $\beta = 0$ and when $\beta \neq 0$, defined as $\text{AF}_{\text{SFA}}^{\text{DLTF}(\beta=0)}$ and $\text{AF}_{\text{SFA}}^{\text{DLTF}}$ respectively, is formalized in Theorem 13.1 and Theorem 13.2, respectively.

THEOREM 13.1. *When applying DLTF for task partitioning, if $\beta = 0$, the approximation factor of DLTF combined with SFA in terms of energy consumption is*

$$AF_{SFA}^{DLTF(\beta=0)} \leq \max\left\{AF_{SFA}^{\beta=0}, \psi_{LTF}^{\gamma-1} \cdot h\left(\frac{4M+1}{6M}\right)\right\}.$$

PROOF. This comes from Pagani and Chen [2013, Lemmas 4 and 6]. □



(a) when $\beta = 0$ (b) when $\beta \neq 0$.

Fig. 12. Approximation factors for SFA combined with DLTF.

THEOREM 13.2. *When applying DLTF for task partitioning, if $\beta \neq 0$, the approximation factor of DLTF combined with SFA in terms of energy consumption is*

$$AF_{SFA}^{DLTF} \leq \max \left\{ AF_{SFA}, \frac{\gamma - 1}{\psi_{LTF}\left[\gamma^{\gamma} h\left(\frac{4M+1}{6M}\right)\right]^{\frac{1}{\gamma-1}}} + \psi_{LTF}^{\gamma-1} \cdot h\left(\frac{4M+1}{6M}\right) \right\}.$$

PROOF. This comes from Pagani and Chen [2013, Lemmas 8 and 9]. □

The first case for both theorems, with values equal to the approximation factor of SFA for a given task partition (presented in Theorem 5.2 and Theorem 6.2), happens when there exists a task with much higher utilization than the rest of the tasks. For a specific γ, after a certain number of cores per island, such a case dominates over the other case in which the tasks can be more fairly partitioned. Furthermore, this effect can be observed in Figure 12(a) and Figure 12(b), where we present the approximation factor for DLTF combined with SFA in terms of energy consumption when $\beta = 0$ and when $\beta \neq 0$, respectively.

14. CONCLUSIONS

To the best of our knowledge, SFA is the state-of-the-art solution for energy efficiency when considering periodic real-time tasks. In fact, SFA has been adopted by several researchers in the past, for example, Devadas and Aydin [2010] and Nikitin and Cortadella [2012], mainly because executing always at a single feasible frequency inside a voltage island is a simple and intuitive scheme. Furthermore, since all the cores run at a single frequency and no frequency alignment for DVFS between cores is needed, any unicore dynamic power management technique for reducing the energy consumption for idling can be easily incorporated. We only focus on the analysis of the approximation factor for the worst cases. The applicability of SFA with slack reclamation to deal with early completion of tasks can be found in Devadas and Aydin [2010].

In this article we have analysed the approximation factor of SFA for energy efficiency. We have shown that the approximation factor can be bounded to a value depending on γ and the number of cores in the voltage island. We also evaluated the lower bound of the approximation factor for SFA by providing case studies with different utilization

distributions. The simulations show that the analytical upper bound is not far away from the peak factor of approximation based on simulations and also that, in general SFA is a good scheme, whereas the worst case only happens under particular utilization distributions. Based on the simulation results for a concrete case study of SCC [Intel 2009], SFA is shown as a good approximation even for such worst cases. In other words, our analysis is for the *worst case* and further practical considerations, such as s_{crit} close to $s_{\max}$ for some platforms with 22nm technology, would make our analysis more pessimistic, further supporting the effectiveness of SFA.

The analysis for SFA for fixed task sets (already mapped to cores) can be used as a cornerstone for task partitioning, as shown by the work from Pagani and Chen [2013] in which the results presented in this article are extended to derive the approximation factor of SFA combined with a task partitioning strategy against the optimal task partitioning and optimal DVFS schedule in terms of energy efficiency.

REFERENCES

Susanne Albers and Antonios Antoniadis. 2012. Race to idle: New algorithms for speed scaling with a sleep-state. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*. 1266–1285.

Hakan Aydin and Qi Yang. 2003. Energy-aware partitioning for multiprocessor real-time systems. In *Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS'03)*. 113–121.

Philippe Baptiste, Marek Chrobak, and Christoph Durr. 2012. Polynomial-time algorithms for minimum energy scheduling. *ACM Trans. Algor.* 8, 3.

Shekhar Borkar. 2007. Thousand core chips: A technology perspective. In *Proceedings of the 44th Design Automation Conference (DAC'07)*. 746–749.

Jian-Jia Chen, Heng-Ruey Hsu, and Tei-Wei Kuo. 2006. Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*. 408–417.

Jian-Jia Chen and Tei-Wei Kuo. 2007. Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'07)*. 289–294.

Jian-Jia Chen and Lothar Thiele. 2010. Energy-efficient scheduling on homogeneous multiprocessor platforms. In *Proceedings of the ACM Symposium on Applied Computing (SAC'10)*. 542–549.

Pepijn J. de Langen and Ben H. H. Juurlink. 2006. Leakage-aware multiprocessor scheduling for low power. In *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS'06)*.

Vinay Devadas and Hakan Aydin. 2010. Coordinated power management of periodic real-time tasks on chip multiprocessors. In *Proceedings of the International Conference on Green Computing (GREENCOMP'10)*. 61–72.

Ronald L. Graham. 1969. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math. 17,* 263–269.

Sebastian Herbert and Diana Marculescu. 2007. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'07)*. 38–43.

Jason Howard, Saurabh Dighe, Sriram R. Vangal, Gregory Ruhl, Nitin Borkar, et al. 2011. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *IEEE J. Solid-State Circ.* 46, 1, 173–183.

Intel. 2009. Single-chip cloud computer (scc). http://www.intel.com/content/www/us/en/research/intel-labs-single-chip-cloud-overview-paper.html.

Sandy Irani, Sandeep Shukla, and Rajesh Gupta. 2003. Algorithms for power savings. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*. 37–46.

Ravindra Jejurikar, Cristiano Pereira, and Rajesh Gupta. 2004. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st Design Automation Conference (DAC'04)*. 275–280.

Chang L. Liu and James W. Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM* 20, 1, 46–61.

Gabriel A. Moreno and Dionisio de Niz. 2012. An optimal real-time voltage and frequency scaling for uniform multiprocessors. In *Proceedings of the 18th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'12)*. 21–30.

Nikita Nikitin and Jordi Cortadella. 2012. Static task mapping for tiled chip multiprocessors with multiple voltage islands. In *Proceedings of the 25th International Conference on Architecture of Computing Systems (ARCS'12)*. Springer, 50–62.

Santiago Pagani and Jian-Jia Chen. 2013. Energy efficient task partitioning based on the single frequency approximation scheme. In *Proceedings of the 34th IEEE Real-Time Systems Symposium (RTSS'13)*. 308–318.

Ronald L. Rardin. 1998. *Optimization in Operations Research*. Prentice Hall.

Euiseong Seo, Jinkyu Jeong, Seon-Yeong Park, and Joonwon Lee. 2008. Energy efficient scheduling of real-time tasks on multicore processors. *IEEE Trans. Parallel Distrib. Syst.* 19, 11, 1540–1552.

Ruibin Xu, Dakai Zhu, Cosmin Rusu, Rami Melhem, and Daniel Mosse. 2005. Energy-efficient policies for embedded clusters. In *Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05)*. 1–10.

Chuan-Yue Yang, Jian-Jia Chen, and Tei-Wei Kuo. 2005. An approximation algorithm for energy-efficient scheduling on a chip multiprocessor. In *Proceedings of the Conference on Design, Automation, and Test in Europe (DATE'05)*. 468–473.

Received September 2013; accepted May 2014