Single Frequency Approximation Scheme for Energy Efficiency on a Multi-core Voltage Island

Technical report: KIT-MTA-2013-0001

Published in RTCSA 2013 under title: ***Energy Efficiency Analysis for the Single Frequency Approximation (SFA) Scheme***

Santiago Pagani and Jian-Jia Chen
Department of Informatics, Karlsruhe Institute of Technology (KIT), Germany
E-mail: santiago.pagani@kit.edu, jian-jia.chen@kit.edu

***Abstract*—Energy-efficient designs are important issues in computing systems. This paper studies the energy efficiency of a simple and linear-time strategy, called Single Frequency Approximation (SFA) scheme, for periodic real-time tasks on multi-core 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, 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 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 uni-core dynamic power management technique for reducing the energy consumption for idling can be easily incorporated individually on each core in the voltage island. This paper also provides the analysis of energy consumption for SFA, combined with the procrastination for Dynamic Power Management (DPM). Furthermore, we also extend our analysis for deriving the approximation factor of SFA for a multi-core system with multiple voltage islands.**

I. 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 has motivated computing systems to move from single-core to multi-core platforms, to balance the power consumption and computation performance.

As shown in the literature, e.g. [1], the dynamic power consumption (mainly generated by switching activities) and the static power consumption (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 non-negligible 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 a combined slowdown and shutdown approach for energy minimization, e.g., in [2], [3], [1].

In the past decade, task scheduling and partitioning have been explored for energy reduction, while the performance requirements can still be met. However, most researches either assume that each core can change its own supply voltage independently from the others, e.g., [4], [5], [6], [7], or consider the other extreme direction where there exists only one global supply voltage for all the cores, which is energy-inefficient.

For the next-generation many-core systems, a trade-off between global-voltage and local-voltage (per-core DVFS) platforms is to adopt multi-core architecture with different voltage islands, in which several cores on a voltage island share the same supply voltage [8], [9]. For example, Intel has released a research multi-core platform, called Single-chip Cloud Computer (SCC) [10], [11], 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 multi-core systems has been widely explored, especially for real-time embedded systems, e.g., [4], [5], [6], [7]. 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 [9] that per-core DVFS suffers from complicated design problems.

For global voltage scaling, the result in [12] provides answers on voltage scaling to minimize the energy consumption, by using an accelerating schedule when the system has *frame-based real-time tasks*, in which all the tasks have the same arrival time and period. However, the approach in [12] is highly restricted and cannot be easily extended to handle periodic real-time tasks, in which tasks have different periodicity, or systems with non-negligible static/leakage power consumption. The study in [13], [14] relaxes the assumptions in [12] by considering periodic real-time tasks with non-negligible static and voltage-independent power consumptions and non-negligible overhead for turning to low-power idle modes. The approach in [13] decides the number of active cores, and then decides the frequency of the active cores. However, there is no theoretical analysis in [13] to show the effectiveness of their approach for energy minimization. The work in [14], 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 multi-core systems with

a shared supply voltage in a voltage island, it is necessary to choose a policy that decides the voltage/frequency for execution. The *simplest* and *most intuitive* strategy is to use a single voltage/frequency for executing, particularly, the lowest voltage/frequency that satisfies the timing constraints. We denote such a scheme as *Single Frequency Approximation* (*SFA*) scheme. After the task partitioning is done (which is not the focus of this paper), SFA has linear time complexity, only from evaluating the task set with the highest cycle utilization.

Even though SFA is not an optimal strategy for energy efficiency, it reduces the management overhead significantly. SFA does not require frequent voltage/frequency changes at run time, as it only requires one frequency. Furthermore, since no frequency alignment for DVFS between cores is needed under SFA, any uni-core 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, e.g., [13] (when tasks do not complete earlier than the estimated worst-case execution times) and [15]. SFA is indeed a good strategy when the workload is *perfectly* balanced, i.e., all the cores are assigned with the same cycle utilization. On the contrary, if the utilizations of the cores are *skewed*, i.e., 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 that 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 multi-core 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 above discussions, the goal of this paper is to provide comprehensive analysis, from a theoretical point of view, to show the effectiveness of SFA for energy minimization, particularly for the 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:

- 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. The analysis is presented from Section III to Section VII.
- Furthermore, in Section VIII, we also show the effectiveness of SFA by analysing the approximation factor when applying SFA together with the uni-core DPM procrastination scheme from [2]. We show that the overall approximation factor by considering such DPM scheme and SFA is increased by 1 from the analysis in Section VI.
- Specifically, in Section IX, 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, in Section X and Section XI, we sketch simple extensions to consider cores with a limited set of available frequencies and systems with multiple voltage islands.

II. System Model and Problem Definition

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

A. Hardware Model

This paper 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, e.g., one voltage island of SCC [10], [11].[1] The system can change the voltage and frequency of the island by adopting DVFS. This model has also been adopted in [12], [13]. 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 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 $P(s)$, and the energy consumption during time interval Δt at frequency s is $E(s) = P(s) \cdot \Delta t$. 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, e.g., [5], [6], [7], [12], [13], where the most widely used power consumption function, which we adopt for this paper, 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 Equation (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 $s_{\text{crit}} = \max\left\{s_{\min}, \sqrt[\gamma]{\frac{\beta}{(\gamma-1)\alpha}}\right\}$. 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 [1], [4].

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

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

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

TABLE I: Experimental results from [11] (48-core system).

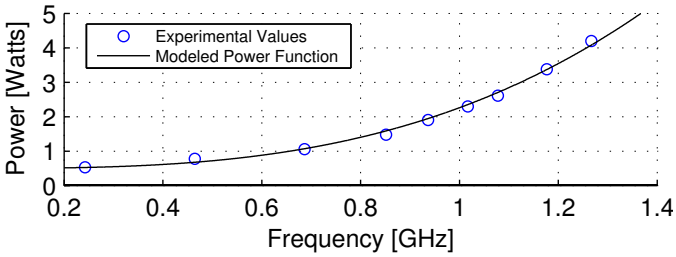

Fig. 1: Power Model from Experimental Results in [11].

Furthermore, we can use the power consumption function from Equation (1) to model the experimental results from [11], in which a multi-core system that integrates 48 cores was developed. Figure 12 (Frequency vs. voltage) and Figure 13 (Measured power vs. voltage) from [11] are of particular interest, and its values are summarized in Table I.

We approximate Table Ia using a quadratic function. Thus, by having a function that relates frequency with voltage, we are able to rewrite Table Ib relating frequency with power. The power values of this new table are divided by 48, since this 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 Equation (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. This 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, an *Adjusted R-square* of 0.9958 and a *Root Mean Squared Error (RMSE)* of 0.07938. Figure 1 shows this power model and the original experimental measurements from [11].

When a core does not execute anything, we consider that it enters a low-power mode with power consumption $\beta' \geq 0$. 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. When a core enters/leaves a low-power mode, we assume negligible overhead in our analysis for Sections IV to VI. For systems with non-negligible overhead, the strategy by considering the *break-even time* and *procrastination schemes*, e.g., in [2], [16], can be further adopted and this is extended in Section VIII.

B. 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 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. When the task instances arrive to the system, it will be 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.

After the task partitioning is done by using M cores, 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 paper and is assumed to be 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, e.g., [17], that executing core i at frequency higher than or equal to w_i with EDF will meet the timing constraint.

The least common multiple (LCM) of the periods of all task is called the hyper-period and is denoted as L.

C. 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 paper is to analyse the approximation factor of SFA, defined AF_{SFA} and expressed as

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

where E_{OPT} is the optimal energy consumption during a hyper-period, E_{SFA} is the energy consumption for SFA during a hyper-period, and E^* is a lower bound for the optimal energy consumption for any feasible schedule during a hyper-period. 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 run time, 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 [13].

III. Lower Bound Energy Consumption

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

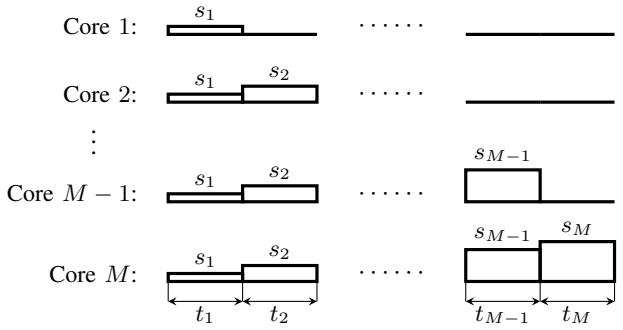

Fig. 2: A schedule satisfying the *deep sleeping property*.

A. Preliminary Results with $\beta = 0$

A special case of the problem is frame-based real-time tasks, in which all the tasks arrive at time 0 and have the same period and deadline. For such case, the period of any task is also the hyper-period L. Yang et al. [12] have proposed a scheme based on the *deep sleeping property* (every core is put to sleep after executing its workload), as shown in Figure 2. For completeness, we summarize the schedule proposed in [12] when the task sets are already assigned onto cores. In [12], 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 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 Equation (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. [12] provide an optimal frequency assignment for the above 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 hyper-period 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. [12], in Equation (5).

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

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

This subsection analyses the lower bound of the energy consumption when β is not negligible. Similar to Section III-A, 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 [12], with $s_{\min} = 0$ and $\beta \neq 0$, Equation (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 [18] on Equation (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 Equation (7), the resulting t_i is equal to $\frac{c_i}{s_{\text{crit}}}$, for all $i = 1, 2, \ldots, M$. Namely, all the cores run at frequency s_{crit} for execution. Clearly, when $w_M \leq s_{\text{crit}}$, the above solution is a feasible one.

For the case that $\sum_{i=1}^{M} t_i = L$, then $\lambda > 0$, which conceptually means that it is no longer feasible to meet the timing constraints by running all cores at s_{crit} for execution. Hence, from Equation (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 Equation (8) is λ. Since Equation (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 Equation (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 1 shows how we estimate E^*, based on an auxiliary frequency defined as s_{dyn}.

Lemma 1: For a given frequency s_{dyn} with $s_{\text{crit}} < s_{\text{dyn}} < s_{\max}$, a safe lower bound for the optimal energy consumption for any feasible schedule can be expressed as

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

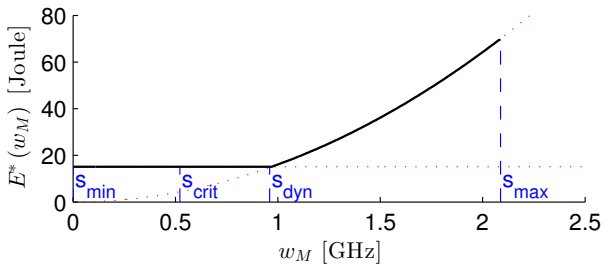

Fig. 3: $E^*(w_M)$ when $\alpha = 1.76\frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts and $M = 20$ with $L = 1$ sec and $c_i = 5 \cdot 10^7$cycles for all $i = 1 \dots M$.

Proof: According to the above analysis and lower bound for the optimal energy consumption in Equation (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 Equation (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 Equation (9) can be found in Figure 3.

IV. Energy Consumption of SFA

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

For periodic real-time tasks, the workload to be completed on core i during the hyper-period is $L \cdot w_i$. From Equation (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 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 Equation (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 above solution is a feasible one. Therefore, *when $w_M \leq s_{crit}$, 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 III and will be considered until Section IX, inclusive. *The approximation factors obtained for this condition are safe upper bounds for the general case.*

Finally, the frequencies chosen by SFA are (1) s_{crit} if w_M is less or equal than s_{crit} and (2) w_M, otherwise. By replacing this s_u values in Equation (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 Equation (11).

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

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

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

V. 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 Equation (5) and the dynamic energy consumption of SFA in Equation (12).

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

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

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

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

where for notational brevity

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

Intuitively, from Equation (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, i.e., $w_1, w_2, \dots, w_{M-1}$, are distributed. However, for the lower bound of energy consumption, the utilization distribution matters. Specifically, we find a *critical utilization distribution* when $w_1 = w_2 = \cdots = w_{M-1} = \frac{\sum_{i=1}^{M-1} w_i}{M-1}$, which results in a lower bound for Equation (5) and Equation (9), and in an upper bound for $H(r_0, \dots, r_M)$. This is formally presented Lemma 2.

Lemma 2: For all $r_0, r_1 \dots, r_M$, defined in Equation (13), and by defining δ as $\frac{\sum_{i=1}^{M-1} r_i}{M-1}$, with $\gamma > 1$, we have

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

Proof: Suppose that we change the configuration from $r_1, \dots, r_{M-1}$ to $r'_1, \dots, r'_{M-1}$ 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 Equation (15), we know that $H(r'_0, \dots, r'_M)$ under fixed $\sum_{i=1}^{M-1} r'_i$ is maximized if and only if $\sum_{i=1}^{M-1}\left(r'_i - r'_{i-1}\right)\sqrt[\gamma]{M-i+1}$ under fixed $\sum_{i=1}^{M-1} r'_i$ is minimized. By using the extreme

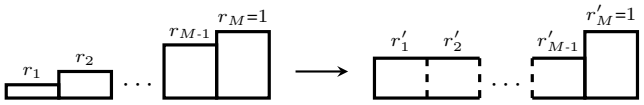

Fig. 4: Resulting Cycle Utilization Relation Change.

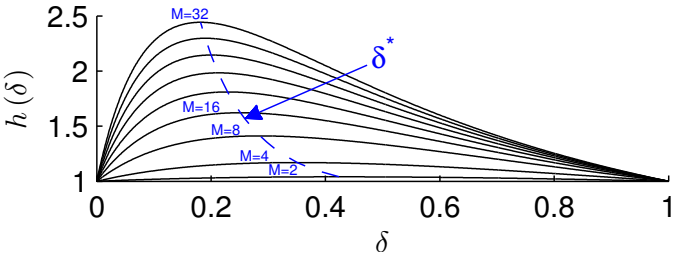

Fig. 5: $h(\delta)$ when $\gamma = 3$, highlighting δ^*.

point theorem, 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 4). By setting r_i to δ for $i = 1, 2, \ldots, M-1$, in Equation (15), the lemma is proven. ■

By taking the first order derivative of $h(\delta)$ with respect to δ, defined in Lemma 2, 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 follows

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

A representation of $h(\delta)$ when γ is 3 can be seen in Figure 5.

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 1.

Theorem 1: When $\beta = 0$, the approximation factor $\text{AF}_{\text{SFA}}^{\beta=0}$ of SFA for periodic real-time tasks is

$$\text{AF}_{\text{SFA}}^{\beta=0} \leq h(\delta^*), \tag{18}$$

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

Proof: Based on the definition of function $h()$ in Lemma 2, the definition of $H(r_0, \ldots, r_M)$ in Equation (15), and the relation between $\text{AF}_{\text{SFA}}^{\beta=0}$ and $H(r_0, \ldots, r_M)$ in Equation (14), we can express $\text{AF}_{\text{SFA}}^{\beta=0}$ as a function of δ^* to obtain the inequality in Equation (18). ■

A representation of $\text{AF}_{\text{SFA}}^{\beta=0}$, for different values of M when $\gamma = 2$ and $\gamma = 3$ can be seen in Figure 6.

VI. Approximation Factor for SFA with $\beta \neq 0$

After deriving the lower bound of energy consumption that considers leakage in Equation (9) and the energy consumption of SFA in Equation (11), this section presents the analysis for the approximation factor, defined in Equation (3), for SFA with

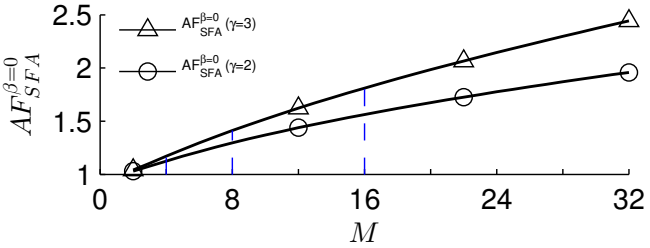

Fig. 6: Approximation factor for SFA with $\beta = 0$.

$\beta \neq 0$. We will first present the analysis based on a given s_{dyn}, as defined in Lemma 1. Then, we will analyse the approximation factor based on the *critical utilization distribution* among the task sets to provide a safe factor.

A. Approximation Factor as a function of s_{dyn}

Replacing the different possible values of E_{SFA} and E^* from Equation (11) and Equation (9) in Lemma 1 respectively, we get the relation $\frac{E_{\text{SFA}}}{E^*}$ as a function of w_M:

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

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

Lemma 3: By defining s_{dyn} as

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

we have

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

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 Equation (13) and the definition of $H(r_0, \ldots, r_M)$ from Equation (15), we can rephrase Equation (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 (22) is a decreasing function with respect to w_M.

An example for the approximation factor function $\frac{E_{\text{SFA}}}{E^*}(w_M)$ can be seen in Figure 7. With the above analysis,

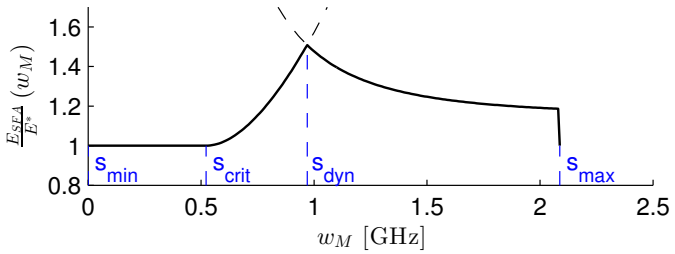

Fig. 7: $\frac{E_{\text{SFA}}}{E^*}(w_M)$ when $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts and $M = 20$ with $L = 1$ sec and $c_i = 5 \cdot 10^7$cycles for all $i = 1 \dots M$.

function $\frac{E_{\text{SFA}}}{E^*}(w_M)$ reaches the (lowest) upper bound when the above two cases intersect with each other. That is, when

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

which results in the definition of s_{dyn} in Equation (20). Thus, Equation (21) holds and this lemma is proven. ■

B. Approximation factor for the critical utilization distribution

Lemma 3 presents the analysis of relation $\frac{E_{\text{SFA}}}{E^*}(w_M)$ for a given cycle utilization distribution, namely, $w_1, w_2, \dots, w_M$. To obtain the approximation factor from Equation (3), we rephrase this relation for the *critical utilization distribution* (also used when $\beta = 0$), presented in Lemma 2.

From Lemma 2, we rewrite Equation (20) as a function of $h(\delta^*)$ and define s^*_{dyn} as the upper bound for the value of s_{dyn}:

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

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

Theorem 2: When $\beta \neq 0$, the approximation factor AF_{SFA} for periodic real-time tasks is

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

where δ^* is defined as a function of γ and M in Equation (17) and $h()$ is defined in Equation (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 Equation (3), based on Equation (21) in Lemma 3, Equation (16) in Lemma 2, and Equation (23) as well as the definitions $s_{\text{crit}}{}^{\gamma} = \frac{\beta}{(\gamma-1)\alpha}$ and function $h()$, we can replace s^*_{dyn} as a function of δ^* to obtain the inequality in Equation (24). ■

VII. Approximation factor for balanced task sets

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

Lemma 4: Given $0 < r_1 \leq r_2 \leq \dots \leq r_M = 1$, $M \geq 2$ 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 Equation (16) (illustrated in Figure 5), it is clear that $h(\delta)$ is a decreasing function with respect of 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 Equation (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 Equation (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 $M = 2$, i.e., 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 $\gamma > 1$, and the lemma is proven. ■

Theorem 3: Given $M \geq 2$, if δ, defined in Lemma 2 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

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

or

$$\text{AF}_{\text{SFA}}(\delta \geq 0.5) \leq \frac{\gamma - 1}{[\gamma\gamma h(0.5)]^{\frac{1}{\gamma-1}}} + h(0.5), \tag{26}$$

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

Proof: This is based on Lemma 4 and Theorem 2. ■

Corollary 1: Clearly, δ plays a major role in the approximation factor of SFA. Even though we do not analyse task partitioning for SFA, from Theorem 3, 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 amount of execution cycles of any task is no more than the average execution cycles on all cores, a load balancer, like the largest-task-first strategy in [12], results in $\frac{w_1}{w_M} \geq 0.5$ (as proved in Lemma 5 in [12]).

VIII. Non-negligible switching overhead

When the energy overhead for entering/leaving a low-power mode is non-negligible, 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*, which is 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 uni-core procrastination algorithm* in the literature to decide when to switch a core to a low-power mode, e.g., [2], [16], and the analysis for combining SFA with each algorithm should be studied accordingly. Particularly, this section analyses the approximation factor of using SFA combined with algorithm Left-to-Right (LTR) from [2], against the optimal DPM solution. That is, we use SFA to decide the voltage/frequency of the island, and use LTR to decide whether/when

For notational brevity, we isolate certain portions of energy consumption for a schedule S, as done in [2]:

- $\text{energy}(S)$: The total energy consumed by schedule S during a hyper-period.
- $\text{active}(S)$: The energy expended while the system is active, i.e., the energy consumption for *executing tasks*.

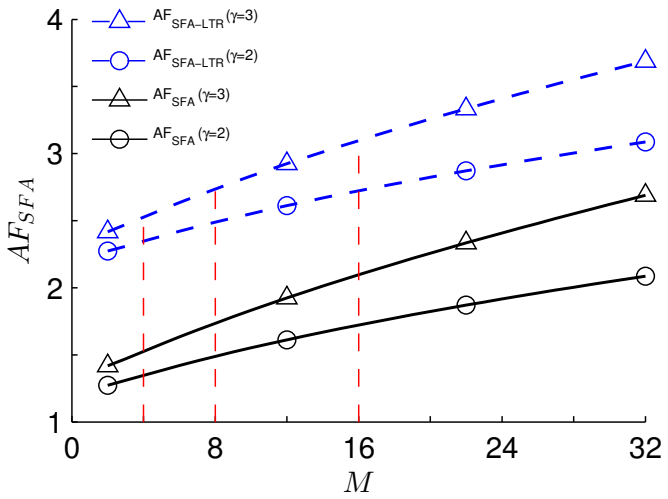

Fig. 8: Approximation factors for SFA.

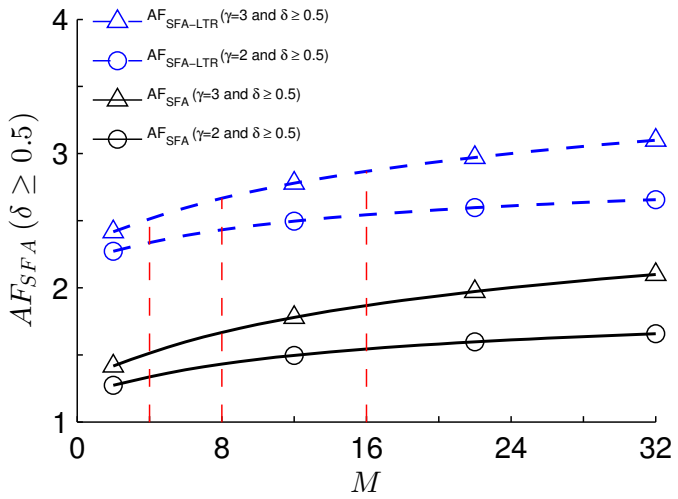

Fig. 9: Approximation factors for SFA with $\delta \geq 0.5$.

- idle (S): 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): The cost to keep the system in the *on* state while the system is on.
- sleep (S): 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 schedule, whereas $S_{\text{OPT},m}$ is the corresponding schedule by considering only the m-th core. Similarly, we define $S_{\text{LTR}}^{\text{SFA}}$ as the schedule by using SFA for executing and LTR for sleeping, whereas $S_{\text{LTR},m}^{\text{SFA}}$ is the corresponding schedule by considering only the m-th core.

According to Theorem 2, we know that

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

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

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

Theorem 4: When $P(s)$ is a convex and increasing function, combining SFA and LTR results in an approximation factor $\text{AF}_{\text{SFA-LTR}}$ equal to $\text{AF}_{\text{SFA}} + 1$.

Proof: From Equation (27) and Equation (28), and considering that by definition $\text{AF}_{\text{SFA}} \geq 1$, we have

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

Hence, the theorem is proven. ■

IX. 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 1 and Theorem 2, the approximation factor of SFA depends on γ and M. Hence, we consider some practical settings for γ, i.e., $\gamma = 2$ and $\gamma = 3$, and we explore the impact of M on the approximation factor. Theoretically, the approximation factor can go up to ∞ when $M \to \infty$. However, practically, 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 8 presents the approximation factor, based on Theorem 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 non-negligible overhead for sleeping, from Theorem 4, this values are incremented by 1.

Motivated by the conclusions from Corollary 1, Figure 9 shows the approximation factor for SFA under the condition $\frac{w_1}{w_M} \geq 0.5$ based on Theorem 3, i.e., $\delta = 0.5$. 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, 1.66, respectively) when $\gamma = 2$. When we consider non-negligible overhead for sleeping, from Theorem 4, this values are incremented by 1.

X. Systems with Discrete Frequencies

This section presents a simple extension for systems with power consumption function $P(s) = \beta + \alpha s^{\gamma}$ and with discrete frequencies $\{f_1, f_2, \ldots, f_F\}$, such that $f_1 = s_{\min}$ and $f_F = s_{\max}$. Due to space limitations, we briefly sketch the basic concept of such an extension. Given the convexity

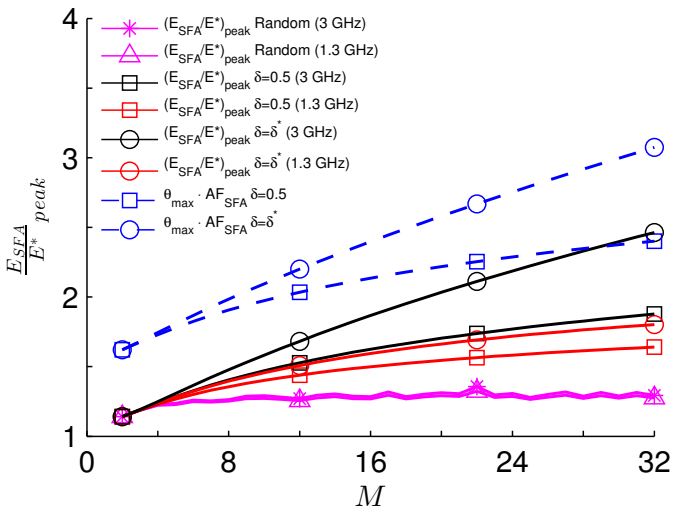

Fig. 10: Simulation results with $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts, $\gamma = 3$ and negligible overhead for sleeping.

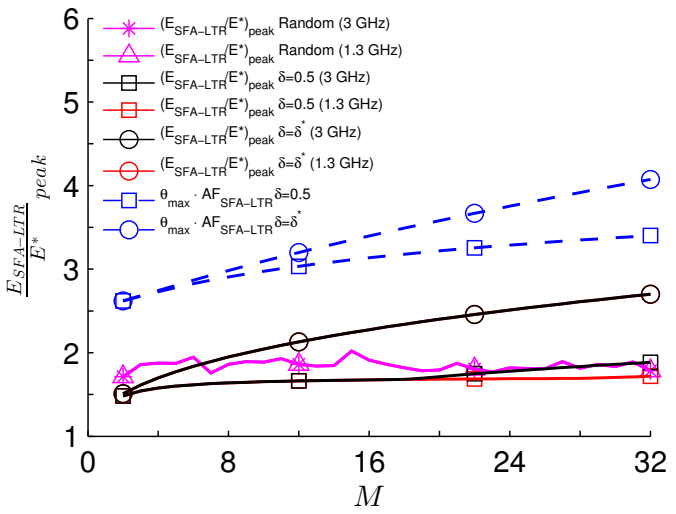

Fig. 11: Simulation results with $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts and $\gamma = 3$ and non-negligible overhead for sleeping.

of $\frac{P(s)}{s}$, the lower bound of the energy consumption by considering continuous frequencies, i.e., E^*, is also a lower bound for the optimal energy consumption under discrete frequencies. When $f_{i-1} < w_M \leq f_i$, the energy consumption of SFA for execution by running at frequency f_i is equal to that (of SFA) at frequency w_M multiplied with $\theta(w_M) = \frac{P(f_i) \cdot w_M}{P(w_M) \cdot f_i}$. Therefore, clearly, the approximation factor of SFA for discrete frequencies is equal to $\text{AF}_{\text{SFA}} \cdot \theta_{\max}$, with $\theta_{\max} = \max_{1<i\leq F} \frac{P(f_i) \cdot f_{i-1}}{P(f_{i-1}) \cdot f_i}$, depending on the hardware parameters and the available frequencies. For example, for a system with $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\beta = 0.5$ Watts, $\gamma = 3$, and available frequencies $\{0.1 \text{ GHz}, 0.2 \text{ GHz}, \ldots, 3.0 \text{ GHz}\}$, the value of $\theta_{\max}$ is equal to 1.14.

XI. 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 multi-core systems with multiple voltage islands as follows:

Theorem 5: For a system with V voltage islands under a given mapping of task partitions, the approximation factor (with respect to the energy consumption) $\text{AF}_{\text{SFA}}^{\text{V-islands}}$ by running each voltage island with SFA is equal to the approximation factor AF_{SFA} of SFA in an individual voltage island.

Proof: The proof comes directly from the definition of AF_{SFA} in Equation (3).

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

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

XII. 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 2, we use Newton's method for solving Equation (8) for a given input instance. Thus, providing a *concrete factor* based on the energy consumption of SFA.

A. 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 [11], as explained in Section II-A. We consider three cases for the cycle utilization distributions of the task sets: (1) the theoretical *critical utilization distribution* from Lemma 2 with $\delta = \delta^*$, (2) the *critical utilization distribution* based on Lemma 2 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 hyper-periods $L = 1, 2, \ldots, 5$ seconds for every w_M. Case (a) corresponds to the practical setting of (not continuous) available frequencies $\{0.1 \text{ GHz}, 0.2 \text{ GHz}, \ldots, 1.3 \text{ GHz}\}$ in the 48-core system from [11], and 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}), 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 $\frac{E_{\text{SFA}}}{E^*}_{\text{peak}}$. 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.

B. Simulation Results

Figure 10 presents the results of $\frac{E_{\text{SFA}}}{E^*}_{\text{peak}}$ for the six configurations, i.e., (1a), (1b), (2a), (2b), (3a) and (3b), together with the analytical upper bounds $\theta_{\max} \cdot \text{AF}_{\text{SFA}}$ derived from

Theorem 2 and Theorem 3 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 this 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* $\theta_{max} \cdot AF_{SFA}$ 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, and this happens not only because of the precise computation of E^* by using Newton's method to solve Equation (8), but because $s_{\max}$ is close to s_{crit} (2.5 times its value). This result, even though it makes our analysis pessimistic for such practical consideration, it further supports the effectiveness of SFA for such platforms, e.g., SCC [10]. Moreover, for cases (3a) and (3b), $\frac{E_{\mathrm{SFA}}}{E^*}_{\mathrm{peak}}$ is no more than 1.5, suggesting that for *general cases* SFA is a simple and effective scheme for energy minimization, and complicated DVFS solutions are not necessary unless better results for the worst-case are required.

Figure 11 presents similar results as Figure 10, but for frame-based tasks using SFA in combination with LTR. The non-negligible 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, which is a lower bound for the optimal energy consumption. The reason why we choose this lower bound is due to the difficulty to derive tighter lower bounds when $\beta \neq 0$ and non-negligible energy overhead for entering/leaving a low-power mode is consider. Especially, such ignorance in our setting also leads to the significantly higher factor $\frac{E_{\mathrm{SFA\text{-}LTR}}}{E^*}_{\mathrm{peak}}$ for cases (3a) and (3b) when $M \leq 6$ in Figure 11. This happens because for such chases, the energy consumption for executing results in low values both for SFA and the lower bound, which are shadowed by only considered 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).

XIII. 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, e.g., [13] and [15], 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 uni-core 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 [13].

In this paper 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 [10], 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, e.g., s_{crit} close to $s_{\max}$ for some platforms with 22 nm technology, would make our analysis more pessimistic, further supporting the effectiveness of SFA.

We believe that the analysis for SFA for fixed task sets (already mapped to cores) can be used as a cornerstone for task partitioning, as explained in Corollary 1. For future researches, we will consider task partitioning and SFA to minimize the overall energy consumption in multiple voltage islands.

References

[1] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proceedings of the Design Automation Conference*, 2004, pp. 275–280.

[2] S. Irani, S. Shukla, and R. Gupta, "Algorithms for power savings," in *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 37–46.

[3] S. Albers and A. Antoniadis, "Race to idle: new algorithms for speed scaling with a sleep state," in *SODA*, 2012, pp. 1266–1285.

[4] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," in *RTAS*, 2006, pp. 408–417.

[5] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *IPDPS*, 2003, pp. 113 – 121.

[6] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé, "Energy-efficient policies for embedded clusters," in *LCTES*, 2005, pp. 1–10.

[7] P. J. de Langen and B. H. H. Juurlink, "Leakage-aware multiprocessor scheduling for low power," in *IPDPS*, 2006.

[8] S. Borkar, "Thousand core chips: a technology perspective," in *DAC*, 2007, pp. 746–749.

[9] S. Herbert and D. Marculescu, "Analysis of dynamic voltage / frequency scaling in chip-multiprocessors," in *ISLPED*, 2007, pp. 38–43.

[10] Intel Corp., "Single-chip cloud computer."

[11] J. Howard and others, "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *J. Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, 2011.

[12] C.-Y. Yang, J.-J. Chen, and T.-W. Kuo, "An approximation algorithm for energy-efficient scheduling on a chip multiprocessor," in *DATE*, 2005, pp. 468–473.

[13] V. Devadas and H. Aydin, "Coordinated power management of periodic real-time tasks on chip multiprocessors," in *Green Computing Conference, 2010 International*, 2010, pp. 61 –72.

[14] E. Seo, J. Jeong, S.-Y. Park, and J. Lee, "Energy efficient scheduling of real-time tasks on multicore processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1540–1552, 2008.

[15] N. Nikitin and J. Cortadella, "Static task mapping for tiled chip multiprocessors with multiple voltage islands," in *ARCS*, 2012, pp. 50–62.

[16] J.-J. Chen and T.-W. Kuo, "Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems," in *ICCAD*, 2007, pp. 289–294.

[17] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[18] R. L. Rardin, *Optimization in Operations Research*. Prentice Hall, 1998.