Energy Efficiency on Multi-core Architectures with Multiple Voltage Islands

Santiago Pagani, *Student Member, IEEE,* Jian-Jia Chen, *Member, IEEE,* and Minming Li, *Member, IEEE*

Abstract—Efficient and effective system-level power management for multi-core systems with multiple voltage islands is necessary for next-generation computing systems. This paper considers energy efficiency for such systems, in which the cores in the same voltage island have to be operated at the same supply voltage level. We explore how to map given task sets onto cores, so that each task set is assigned and executed on one core and the energy consumption is minimized. Due to the restriction to operate at the same supply voltage in a voltage island, different mappings will result in different energy consumptions. By using the simple Single Frequency Approximation Scheme (SFA) to decide the voltages and frequencies of individual voltage islands, this paper presents the approximation factor analysis (in terms of energy consumption) for simple heuristic algorithms, and develops a dynamic programming algorithm, which derives optimal mapping solutions for energy minimization when using SFA. We experimentally evaluate the running time and energy consumption performance of these algorithms on Intel's Single-chip Cloud Computer (SCC). Moreover, we conduct simulations for hypothetical platforms with different number of voltage islands and cores per island, also considering different task partitioning policies.

Keywords—Energy Efficiency, Multiple Voltage Islands, Single-chip Cloud Computer, Single Frequency Approximation (SFA), DYVIA

1 Introduction

FOR modern computing systems, energy and peak temperature reduction have become important issues to cut the power bills in server systems, to prolong the battery lifetime of embedded systems, or to reduce the packaging cost. Moreover, to balance the power consumption and the computation performance, multi-processor systems on chip and multi-core platforms have been widely applied for modern computing systems.

In the past decade, task scheduling and partitioning have been explored in academia and industry for energy reduction, while the performance indexes can still be met. However, most researches either assume individual voltages for the cores, e.g., [1], [3], [5], [17], or assume a global voltage for all the cores [6], [11], [16], [18]. Providing only one global supply voltage for all the cores is energy-inefficient, whereas providing individual supply voltages for each core locally can be energy-efficient but costly for implementation. Moreover, based on VLSI circuit simulations, it has been suggested in [7] that per-core Dynamic Voltage and Frequency Scaling (DVFS) suffers from complicated design problems.

For the next-generation many-core systems, a trade-off between global-voltage and local-voltage (known as per-core voltage scaling or per-core DVFS) platforms is to adopt multi-core architecture with different voltage islands, in which several cores in a voltage island share the same supply voltage [2], [7] (the cores in a voltage island are naturally consolidated as a cluster). For example, Intel has recently released a research multi-core platform with such a feature, called Single-chip Cloud Computer (SCC) [9] shown in Figure 1. Effective and efficient management of such clusters would be an ultimate objective for the next-generation systems [2], [7].

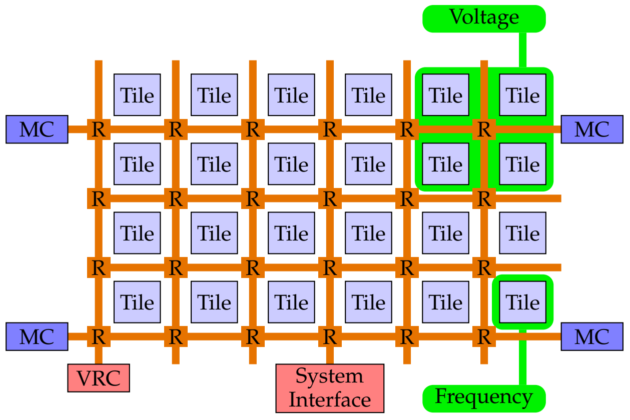


Fig. 1: Intel's Single-chip Cloud Computer Architecture [9], with two IA (P54C) cores per tile, one router (R) associated with each tile, four memory controllers (MC) for off-die (but on-board) DDR3 memory, and a Voltage Regulator Controller (VRC) to set the voltage of islands and frequencies of cores.

Motivations: Even though hardware platforms of multi-core systems with different voltage islands have

- *Santiago Pagani is with the Department of Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. E-mail: santiago.pagani@kit.edu*
- *Jian-Jia Chen is with the Department of Informatics, TU Dortmund University, Dortmund, Germany. E-mail: jian-jia.chen@cs.uni-dortmund.de*
- *Minming Li is with the Department of Computer Science, City University of Hong Kong (CityU), Hong Kong, China. E-mail: minming.li@cityu.edu.hk*

been developed in academia, e.g., [4], [7], [8], [10], [14], and industry, e.g., Intel [9], adopting existing system-level power management schemes, that do not consider the characteristics of different voltage islands, may not work well. When using multiple voltage islands, tasks on the same island are forced to be executed at the same supply voltage (at any given time point), whereas different islands might still use different supply voltages. To meet the timing constraints, a particular task set (group of tasks assigned together on one core) has to be executed at a certain frequency/voltage. When several task sets are mapped onto the same island, one of them may dominate the required supply voltage and significantly increase the energy consumption of the other task sets. Moreover, mapping tasks to a voltage island enforces the activation of the island, which might consume more energy due to the static/leakage power consumption.

As a result, with multiple voltage islands, task set mapping and island activations have to be done carefully to minimize the energy consumption. To the best of our knowledge, there is little research regarding multi-core architectures with multiple voltage islands when the hardware platform is fixed, i.e., when the number of voltage islands is not decided based on the applications. The closest research for mapping can be found in [13], where the *Extremal Optimization* meta-heuristic is applied to solve a similar problem. In [13], a task graph and communication costs between cores are considered, but only one task can be assigned to each core and no theoretical analysis for performance is provided.

Objective: Motivated by the above discussions, it is important to consider the mapping/assignment of given task sets of periodic tasks onto cores that belong to different voltage islands, where the hardware platform, e.g., SCC [9], is given (that is, the number of voltage islands and cores per voltage island is fixed). For such systems, the objective of this paper is to map each task set onto an individual core so that the energy consumption is minimized. We denote the studied problem as the *multiple voltage islands assignment* problem.

Related Work: For per-core voltage scaling, power-aware and energy-efficient scheduling for homogeneous multiprocessor systems has been widely explored, e.g., for real-time embedded systems, [1], [3], [5], [17]. It has been shown in [3] that applying the *Largest-Task-First* (LTF) strategy for task mapping results in solutions with approximation factors, compared to the optimal solutions, 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. [17] and Chen et al. [3] propose polynomial-time algorithms to derive task mappings that try to execute at a *critical frequency*. For homogeneous multiprocessor systems with discrete voltage levels, de Langen and Juurlink [5] provide heuristics for energy-aware scheduling.

For global voltage scaling, the research in [18] provides answers on voltage scaling to minimize the energy consumption. However, the approach in [18] is highly restricted by its assumptions on negligible static/leakage power consumption. Moreover, the mode transition in [18] is also assumed negligible. The study in [6], [16] has relaxed the assumptions in [18] 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 analysis in [15] derives the worst-case approximation factor, in terms of energy consumption, for the *Single Frequency Approximation Scheme* (SFA) for sets of periodic real-time tasks. SFA is a simple strategy that uses a single voltage and frequency for all the cores in a voltage island, to just meet the timing constraints (or the performance requirements).

Our Contributions: For periodic tasks, by adopting SFA in individual voltage islands, our contributions are:

- We present, in Section 4, some simple and intuitive polynomial-time heuristics to map given task sets onto cores in different voltage islands and analyse the approximation factor of *any mapping heuristic* against the optimal solution.
- In Section 6, we provide a dynamic programming algorithm for given task sets, which derives optimal mapping solutions for minimizing the energy consumption when using SFA in individual islands. The time complexity of the dynamic programming algorithm is polynomial when either the number of voltage islands or the number of cores per island is a constant. Moreover, we provide analysis for comparing our algorithm against the optimal mapping solution with the ideal DVFS scheme per voltage island.
- We experimentally evaluate on SCC the running time and energy consumption performance of some mapping heuristics and our dynamic programming solution. Even though the dynamic programming requires more time complexity than the heuristics to derive optimal solutions, the evaluations show that the running time of the algorithm is a few milliseconds for the practical settings when considering up to 6 voltage islands and 8 cores per island, e.g., SCC. This is presented in Section 7.
- Finally, in Section 8, we conduct simulations for hypothetical platforms with several combinations for the number of voltage islands and the number of cores per island, as well as richer DVFS and DPM features than SCC. We also consider different policies for the *Largest-Task-First* (LTF) strategy for task partitioning.

2 System Model and Problem Definition

This section reviews the task and power models adopted in this paper and defines the studied problem. A table summarizing all the symbols used in the paper is included in Appendix A.

2.1 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 c_j. We consider partitioned scheduling, in which each task is assigned onto one core and executed on the assigned core once the task arrives to the system. 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.

For the task partitioning stage, this paper considers the *Largest-Task-First* (LTF) strategy from [18], described in Section 3 for completeness. After the task partitioning has been done by considering M' cores, the tasks are grouped into M' sets of tasks, i.e., $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{M'}\}$. *If there are M homogeneous cores in the system, in order to provide a feasible mapping, it should hold that $M' \leq M$; otherwise, there would be more task sets to map than available cores.*

We define the *cycle utilization* w_i (in *cycles per second*) as $w_i = \sum_{\tau_j \in \mathbf{T}_i} \frac{c_j}{p_j}$. It has been well studied, e.g., [12], that executing task set $\mathbf{T}_i$ at frequencies higher than or equal to w_i with EDF will meet the timing constraints. Moreover, we define the hyper-period D, which is the least common multiple (LCM) of the given task periods for all the tasks.

Without loss of generality, if for the task partitioning $M' < M$, we create $M - M'$ dummy (empty) task sets with zero utilization for the simplicity of presentation, resulting in task sets $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M\}$. Additionally, we consider all the task sets (the given M' non-empty task sets along with the $M - M'$ empty task sets) ordered with respect to their cycle utilizations such that $w_1 \leq w_2 \leq \cdots \leq w_M$. If a task leaves the system or a new task arrives, then we should obtain a new task partition with LTF and the algorithms presented in Section 4 and Section 6 should be re-executed.

For Sections 4, 5 and 6, we will consider the mapping of M' already partitioned task sets together with the corresponding $M - M'$ dummy task sets, i.e., $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M\}$. Moreover, to show the effects of task partitioning combined with the voltage island assignment algorithms (presented in Section 4 and Section 6), in Section 8 we conduct simulations for different task partitioning policies of LTF, i.e., different values for M'.

2.2 Hardware Model

This paper focuses on a system with multiple homogeneous cores and multiple voltage islands, where all the cores in each individual island have to run at the same supply voltage at any given time point, e.g., SCC [9]. Each individual island can change its own voltage and each core can change its own frequency. The combined technique is denoted as Dynamic Voltage and Frequency Scaling (DVFS). For a core to support a frequency, a minimum voltage is required for stable execution.

The system has a number V of voltage islands defined as $\{I_1, I_2, \ldots, I_V\}$, and each island supplies the voltage for Q cores. There are M homogeneous cores in the system, such that $Q \cdot V = M$. We consider that any island consumes negligible power when in the *inactive* state and that it consumes η amount of power when in the *active* state (since there is no voltage regulator with 100% efficiency). We consider an island to be *inactive*, if after the assignment onto cores all the task sets in the island have zero utilization; otherwise it is considered to be *active*. Without loss of generality, the islands are ordered with respect to their voltage in an increasing manner, such that I_1 (I_V) is the island with the lowest (highest) voltage.

Some notations are necessary to identify the task sets assigned to each island. We define set $\mathbf{L}_j = \{\ell_{j,1}, \ell_{j,2}, \ldots, \ell_{j,Q}\}$ as the indexes of the task sets assigned to island I_j such that $\ell_{j,1} < \ell_{j,2} < \cdots < \ell_{j,Q}$ for all $j = 1, 2, \ldots, V$, and the value of $\ell_{j,i} \in [1, M]$ and it is unique for all j, i. Given that the task sets are ordered with respect to their cycle utilizations, it holds that $w_{\ell_{j,1}} \leq w_{\ell_{j,2}} \leq \cdots \leq w_{\ell_{j,Q}}$. For example, if Q is 3 and task sets $\mathbf{T}_5$, $\mathbf{T}_8$, and $\mathbf{T}_9$ are assigned to island I_4, then $\mathbf{L}_4$ is set to $\{5, 8, 9\}$, i.e., $\ell_{4,1} = 5$, $\ell_{4,2} = 8$, and $\ell_{4,3} = 9$.

Once the task sets have been assigned onto cores in different islands, a DVFS policy has to be adopted to decide the execution frequencies for the cores and the voltages for the individual islands. For periodic tasks, a simple and effective strategy is to use a single voltage and frequency for execution, denoted as *Single Frequency Approximation Scheme* (SFA) [15]. That is, with SFA, all the tasks assigned to voltage island I_j will be executed at single frequency s_j (in *cycles per second*), which is no less than $w_{\ell_{j,Q}}$, such that all the task sets assigned to this island meet their timing constraints. The supply voltage of island I_j will be set to the lowest available value, at which all cores in the island can stably execute at frequency s_j. As stated in [15], under SFA, it is guaranteed that EDF can feasibly meet the timing constraints. Without loss of generality, the maximum cycle utilization among all task sets, i.e., w_M, is no more than the maximal core frequency, denoted as $s_{\max}$; otherwise, there is no feasible solution for such task partition.

We consider a general power consumption model for individual cores. We denote the power consumption of a core executing a certain task at frequency s as $P(s)$. The available frequencies are in the range of $[s_{\min}, s_{\max}]$.[1] When a core does not execute anything, we consider that it enters a low-power mode with power consumption $\beta' \geq 0$. That is, when an island is active, the minimum power that any core in the island can consume is β'. This means that when an island is active, it not only consumes η power for being active, but there is also an offset of $Q\beta'$ power consumed by the cores in the island. As we can transfer the power consumption $Q\beta'$ to the power consumption η, without loss of generality, we can set η as $\eta + Q\beta'$ and $P(s)$ as $P(s) - \beta'$, such that we can disregard the effect of the power consumption of a core in a low-power mode, because it is already considered inside the new η. The overheads of entering/leaving a

1. All algorithms still work for systems with discrete frequencies, as shown in the experiments of Section 7 and the simulations of Section 8.

low-power mode are considered negligible.

For analysing the quality of the proposed algorithms in terms of energy minimization, we assume that $P(s)$ is a convex and increasing function with respect to s, and $\frac{P(s)}{s}$ is non-decreasing with respect to s in the range of $[s_{\text{crit}}, s_{\max}]$, where s_{crit} is called the *critical frequency*, representing the frequency that minimizes the energy consumption when the overhead for sleeping is considered negligible. This assumption complies with most of the power models for CMOS processors adopted in the literature, e.g., [1], [3], [5], [6], [17], [18], where the most widely used power consumption function is $P(s) = \alpha s^{\gamma} + \beta$ (with $\alpha > 0$, $\gamma > 1$, and $\beta \geq 0$) and $s_{\text{crit}} = \max\left\{s_{\min}, \sqrt[\gamma]{\frac{\beta}{(\gamma-1)\alpha}}\right\}$.

The lemmas and theorems presented in this paper are only based on the above assumptions for $P(s)$ and $\frac{P(s)}{s}$, and therefore apply for several power models making the obtained results quite general. Specific power functions are only used in this paper when numerical results are required.

2.3 Problem Definition

For the above models, using SFA in individual islands after assigning the task sets onto cores, all cores in island I_j will use the single frequency s_j for execution, which will be set to $max\left\{s_{\text{crit}}, w_{\ell_{j,Q}}\right\}$. The voltage of island I_j will be set to the lowest available value, such that all cores in the island can stably execute at frequency s_j. The energy consumption (during a hyper-period) of the core, in island I_j, where $\mathbf{T}_i$ is assigned, is $P(s_j)\frac{w_i}{s_j}D$. Therefore, for all Q cores in the island, the energy consumption of island I_j during a hyper-period is

$$E_j = \begin{cases} 0 & \text{if } \sum_{i=1}^{Q} w_{\ell_{j,i}} = 0 \\ D \cdot \left(\eta + \frac{P(s_j)}{s_j} \sum\limits_{i=1}^{Q} w_{\ell_{j,i}}\right) & \text{otherwise.} \end{cases} \tag{1}$$

Given partitioned task sets of periodic tasks and a fixed hardware platform with V voltage islands and Q cores per island (V and Q are constants), e.g., SCC [9], the *multiple voltage islands assignment problem* is to map each task set, i.e., $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M\}$, onto an individual core so that the energy consumption is minimized by using SFA on individual islands.

3 Largest-Task-First (LTF) Strategy

For completeness, this section describes the *Largest-Task-First* (LTF) strategy from [18]. LTF is a good and widely used algorithm for task partitioning, with time complexity $O(N(\log N + \log M') + M')$. LTF partitions N tasks $\{\tau_1, \tau_2, \ldots, \tau_N\}$ into M' groups of task sets $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{M'}\}$. As stated in Section 2.2, to feasibly schedule this task partition, the maximum cycle utilization among all task sets should be no more than the maximal core frequency $s_{\max}$. The pseudo-code for LTF is presented in Algorithm 1.

As mentioned in Section 2.1, it should hold that $M' \leq M$; otherwise, there would be more task sets to map than available cores. However, depending on the influence of static/leakage power in $P(s)$ and the value of η, using less than M cores for task partitioning, shutting down cores and voltage islands, could result in energy savings.

Algorithm 1 Largest-Task-First (LTF) strategy

```
Input: Number of task sets $M'$, and tasks $\{\tau_1, \tau_2, \ldots, \tau_N\}$;
Output: Task sets $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{M'}\}$;
1: Sort all tasks in a non-increasing order of their cycle utilizations;
2: for $i = 1$ to $M'$ do
3:     $\mathbf{T}_i \leftarrow \emptyset$;
4: end for
5: for $j = 1$ to $N$ do
6:     Find the task set $\mathbf{T}_i$ with the smallest $w_i$;
7:     $\mathbf{T}_i \leftarrow \mathbf{T}_i + \{\tau_j\}$;
8: end for
9: Re-order $\mathbf{T}_i$ by a non-decreasing order of their cycle utilization;
10: Return $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{M'}\}$;
```

4 Simple Heuristic Algorithms

This section presents two *simple* and *very intuitive* algorithms to solve the problem stated in Section 2.3. It also describes an algorithm extended from [13], based on extremal optimization. These three algorithms have low time complexity but derive non-optimal solutions. This section also provides theoretical analysis for the approximation factor of *any* task partition mapping heuristic against the optimal assignment, under SFA.

4.1 Description of Simple Heuristic Algorithms

Consecutive Cores Heuristic (CCH): One simple and very intuitive heuristic algorithm is to assign the task sets onto cores in each island in a consecutive manner with respect to their cycle utilization. That is, for voltage island I_j with $j = 1, 2, \ldots, V$, we assign the task sets with indexes inside set $\mathbf{L}_j = \{(j-1)Q+1, (j-1)Q+2, \ldots, (j-1)Q+Q\}$ to island I_j. This algorithm has linear time complexity $O(QV)$. For example, if $V = 4$ and $Q = 3$, CCH will assign task sets $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$ to island I_1, task sets $\{\mathbf{T}_4, \mathbf{T}_5, \mathbf{T}_6\}$ to island I_2, task sets $\{\mathbf{T}_7, \mathbf{T}_8, \mathbf{T}_9\}$ to island I_3, and task sets $\{\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}\}$ to island I_4, as shown in Figure 2.

Balanced Utilization Heuristic (BUH): Another simple heuristic algorithm is to assign Q consecutive task sets onto cores in island I_j, such that the difference between the lowest and highest utilization task sets in the island is minimized. That is, if there are still more than Q task sets to be assigned, we find the index i^* such that $w_{i^*+Q-1} - w_{i^*}$ is minimized. Then, these Q task sets are mapped onto one voltage island. By removing these task sets from the problem instance, relabelling the remaining task sets, and repeating the process until there is no unassigned task set left, we can find a feasible mapping. As finding the index i^* requires $O(M + Q)$

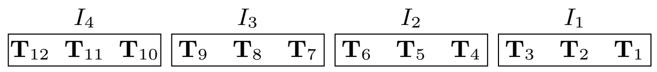

I_4
I_3
I_2
I_1
$\mathbf{T}_{12}$ $\mathbf{T}_{11}$ $\mathbf{T}_{10}$
$\mathbf{T}_9$ $\mathbf{T}_8$ $\mathbf{T}_7$
$\mathbf{T}_6$ $\mathbf{T}_5$ $\mathbf{T}_4$
$\mathbf{T}_3$ $\mathbf{T}_2$ $\mathbf{T}_1$

Fig. 2: Example of CCH for $V = 4$ and $Q = 3$.

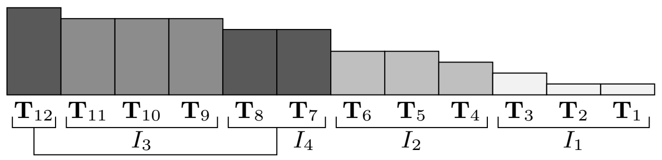

$\mathbf{T}_{12}$ $\mathbf{T}_{11}$ $\mathbf{T}_{10}$ $\mathbf{T}_9$ $\mathbf{T}_8$ $\mathbf{T}_7$ $\mathbf{T}_6$ $\mathbf{T}_5$ $\mathbf{T}_4$ $\mathbf{T}_3$ $\mathbf{T}_2$ $\mathbf{T}_1$
I_3
I_4
I_2
I_1

Fig. 3: Example of BUH for $V = 4$ and $Q = 3$ (the high of the bars represents the cycle utilization of the task sets). Islands are numbered with respect to their voltage, as stated in Section 2.2, and not by the order in which the heuristic assigns the task sets to each island.

time complexity and the number of iterations is at most V, the overall time complexity is $O\left(QV^2\right)$. An example illustrating this algorithm is shown in Figure 3, where $\{\mathbf{T}_9, \mathbf{T}_{10}, \mathbf{T}_{11}\}$ are first assigned to I_3, then $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$ are assigned to I_1, then $\{\mathbf{T}_4, \mathbf{T}_5, \mathbf{T}_6\}$ are assigned to I_2, and finally $\{\mathbf{T}_7, \mathbf{T}_8, \mathbf{T}_{12}\}$ are assigned to I_4.

Extremal Optimization Heuristic (EOH): This algorithm, extended from [13], performs a random search to improve the energy consumption, starting from an arbitrary initial solution, e.g., CCH. The algorithm is based on swapping, by selecting two task sets to swap per iteration: an unfavourable task set and a replacement task set. EOH uses information about the system cost when selecting the swapped task sets, by considering two fitness functions and a power-law distribution. It is expected that such an approach results into fast progress towards the final solution. In addition, EOH accepts new solutions unconditionally, thus making the algorithm easier to tune and avoiding a local minimum. The randomness process is repeated with a predefined number of iterations until there is no improvement. From all the evaluated mappings, EOH returns the one that results in the lowest energy consumption. *Note that EOH is not a contribution of this paper. We present it here for completeness, because we use it for comparisson in Sections 7 and 8.*

The difference between EOH (in this paper) and the algorithm presented in [13], is that we map one task set to one core, instead of just one task per core; and that we do not consider the energy consumed for communication between tasks, thus our fitness functions are simplified. *Although EOH may help to improve the solutions, in the worst cases, the quality of the solution remains as the initial solution.*

4.2 Approximation Factor for Simple Heuristics

This subsection details the proof of the approximation factor for *any* task partition mapping heuristic that uses $\left\lceil \frac{M'}{Q} \right\rceil$ voltage islands with non-empty task sets under SFA (to decide the execution voltage/frequencies of individual islands), against the optimal assignment that also uses SFA in individual islands. That is, although some heuristics might perform well in general cases, for *any* mapping heuristic there exist at least one (worst-case) task partition which results in the approximation factor presented in this subsection.

We denote $E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=ANY}}$ as the energy consumption of island I_j that uses *any* task partition mapping algorithm (ASG=ANY) and SFA to decide the execution voltage/frequencies of individual islands (RUN=SFA). For example, when using heuristic CCH, $E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=ANY}}$ represents the energy consumption on island I_j when mapping the task sets with heuristic CCH and using SFA to decide the execution voltage/frequency on the island. The same holds when considering *any* heuristic. Thus, we do not constraint the analysis using, e.g., $E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=CCH}}$ as notation.

From the definition of set $\mathbf{L}_j = \{\ell_{j,1}, \ell_{j,2}, \ldots, \ell_{j,Q}\}$ in Section 2.2, $\mathbf{T}_{\ell_{j,i}}$ is the task set assigned onto the i-th core of island I_j. We denote $E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=SFA}}$ as the energy consumption of the task sets $\mathbf{T}_{\ell_{j,i}}$ for $i = 1, 2, \ldots, Q$ in the optimal assignment solution under SFA in individual islands (ASG=SFA). Moreover, let $E_{j}{}^{\text{RUN=p.c.DVFS}}_{\text{ASG=ANY}}$ be the energy consumption of the task sets $\mathbf{T}_{\ell_{j,i}}$ for $i = 1, 2, \ldots, Q$ by using per core DVFS. Clearly, $E_{j}{}^{\text{RUN=p.c.DVFS}}_{\text{ASG=ANY}}$ is the lower bound for the energy consumption, since having per-core DVFS is the optimal solution and the task set assignment plays no role for such a case.

Therefore, the approximation factor $\text{AF}^{\text{RUN=SFA}}_{\text{ASG=ANY}}$ is

$$\text{AF}^{\text{RUN=SFA}}_{\text{ASG=ANY}} = \frac{\sum_{j=1}^{V} E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=ANY}}}{\sum_{j=1}^{V} E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=SFA}}} \leq \frac{\sum_{j=1}^{V} E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=ANY}}}{\sum_{j=1}^{V} E_{j}{}^{\text{RUN=p.c.DVFS}}_{\text{ASG=ANY}}}$$
$$\leq \max_{j=1,2,\ldots,V} \left\{ \frac{E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=ANY}}}{E_{j}{}^{\text{RUN=p.c.DVFS}}_{\text{ASG=ANY}}} \right\}.$$

Throughout the proof, we implicitly consider that η is 0, as the optimal solution also requires to use at least $\left\lceil \frac{M'}{Q} \right\rceil$ voltage islands. That is, the worst-case for the approximation factor happens when η is 0. The energy consumption function for $E_{j}{}^{\text{RUN=SFA}}_{\text{ASG=ANY}}$ is defined in Equation (1), and the energy consumption function for $E_{j}{}^{\text{RUN=p.c.DVFS}}_{\text{ASG=ANY}}$ (with $\eta = 0$) is

$$E_{j}{}^{\text{RUN=p.c.DVFS}}_{\text{ASG=ANY}} = D \sum_{i=1}^{Q} \frac{P\left(s_{j,i}\right)}{s_{j,i}} w_{\ell_{j,i}}, \tag{2}$$

such that $s_{j,i} = \max\left\{s_{\text{crit}}, w_{\ell_{j,i}}\right\}$.

Therefore, the approximation factor is expressed as

$$\text{AF}^{\text{RUN=SFA}}_{\text{ASG=ANY}} \leq \max_{j=1,2,\ldots,V} \left\{ \frac{\frac{P(s_{j,Q})}{s_{j,Q}} \sum_{i=1}^{Q} w_{\ell_{j,i}}}{\sum_{i=1}^{Q} \frac{P(s_{j,i})}{s_{j,i}} w_{\ell_{j,i}}} \right\}$$
$$\leq \max_{j=1,2,\ldots,V} \left\{ \frac{1 + \sum_{i=1}^{Q-1} \frac{w_{\ell_{j,i}}}{w_{\ell_{j,Q}}}}{1 + \sum_{i=1}^{Q-1} \frac{s_{j,Q}}{P(s_{j,Q})} \frac{P(s_{j,i})}{s_{j,i}} \frac{w_{\ell_{j,i}}}{w_{\ell_{j,Q}}}} \right\}.$$

The worst case for this relation, regardless of the absolute value of the cycle utilizations and island j, only depends on the ratios $\frac{w_{\ell_{j,i}}}{w_{\ell_{j,Q}}}$ for each island. The maximal

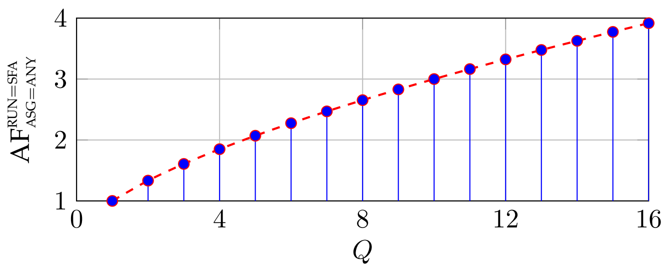


Fig. 4: Approximation Factor for *any mapping* that uses $\left\lceil \frac{M'}{Q} \right\rceil$ islands under SFA, when $P(s) = \alpha s^3$ with $\alpha > 0$.

value is found when $\sum_{i=1}^{Q-1} \frac{w_{\ell_{j,i}}}{w_{\ell_{j,Q}}}$ remains constant and $\sum_{i=1}^{Q-1} \frac{s_{j,Q}}{P(s_{j,Q})} \frac{P(s_{j,i})}{s_{j,i}} \frac{w_{\ell_{j,i}}}{w_{\ell_{j,Q}}}$ is minimized. Since the latter is a convex and increasing function regarding to $\frac{w_{\ell_{j,i}}}{w_{\ell_{j,Q}}}$, it is minimized when $w_{\ell_{j,i}} = \frac{1}{Q-1} \sum_{i=1}^{Q-1} w_{\ell_{j,i}}$ for all $i = 1, 2, \ldots, Q-1$, i.e., when the $Q-1$ least loaded task sets in an island have the same cycle utilization. By defining $x = \frac{1}{Q-1} \sum_{i=1}^{Q-1} \frac{w_{\ell_{j,i}}}{w_{\ell_{j,Q}}}$ as the average cycle utilization of the $Q-1$ least loaded task sets, the approximation factor is

$$AF^{RUN=SFA}_{ASG=ANY} \le \max_{0 \le x \le 1} \frac{1 + (Q-1)\,x}{1 + (Q-1) \frac{s_{j,Q}}{P(s_{j,Q})} \frac{P(s_x)}{s_x} x}, \quad (3)$$

where $0 \le x \le 1$ and $s_x = \max\left\{s_{\text{crit}}, x \cdot w_{\ell_{j,Q}}\right\}$. The specific value of x that maximizes Equation (3) depends on the power consumption function of the cores.

4.3 Numerical Examples for the Approximation Factor of Simple Heuristics

Based on Equation (3), we know the approximation factor for using CCH and BUH. To demonstrate the worst-case behaviour in Equation (3), we provide numerical results based on the specific power consumption function $P(s) = \alpha s^3$. For this case, Equation (3) is re-written as

$$AF^{RUN=SFA}_{ASG=ANY} \le \max_{0 \le x \le 1} \frac{1 + (Q-1)\,x}{1 + (Q-1)\,x^3} \qquad \text{if } P(s) = \alpha s^3. \quad (4)$$

To find the value of x that results in the worst-case, denoted as x_{max}, we set to zero the first order derivative of Equation (4) with respect to x. Hence, we get that $2(Q-1)\,x_{\text{max}}^3 + 3x_{\text{max}}^2 - 1 = 0$, and we simply compute the value of x_{max} for each given Q. Figure 4 presents the resulting approximation factor in Equation (4), for $x = x_{\text{max}}$. The approximation factor is not bounded with respect to Q, and can be very big when Q is very large.

The *tightness* of this analysis can be proven by a concrete example built according to the above conditions. In other words, for a given Q, V, and w_M, if $P(s) = \alpha s^3$, the worst case happens when we have $(Q-1)\,V$ task sets with cycle utilization zero and $Q-1$ task sets with cycle utilization $x_{\text{max}} \cdot w_M$, i.e., $w_M, w_{M-1} = w_{M-2} = \cdots = w_{M-Q+1} = x_{\text{max}} \cdot w_M, w_{M-Q} = w_{M-Q-1} = \cdots = w_1 = 0$. For example, consider a system with $V = 8$, $Q = 8$, $\eta = 0$, and $P(s) = 2 \frac{\text{Watts}}{\text{GHz}^3} \cdot s^3$. After the task partitioning stage, the task sets to be assigned to the islands have cycle utilizations $w_{64} = 10^9, w_{63} = w_{62} = \cdots = w_{57} = 3.544 \cdot 10^8, w_{56} = w_{55} = \cdots = w_1 = 0$. Both CCH and BUH will assign task sets $\mathbf{T}_{57}, \mathbf{T}_{58}, \ldots, \mathbf{T}_{64}$ to island I_8, which according to Equation (1) results in an energy consumption under SFA of 6.96 Joule. The optimal solution however, will assign task set $\mathbf{T}_{57}$ to island I_1, $\mathbf{T}_{58}$ to island I_2, …, and $\mathbf{T}_{64}$ to island I_8, which results in an energy consumption of 2.62 Joule (equivalent energy for having a per-core DVFS platform in this example). The ratio between these two energy consumptions is 2.65, which corresponds to Equation (4) when Q is 8, since for such a case we have that $x_{\text{max}} = 0.3544$.

This example can be easily extended for any value of Q, such that $V \ge Q$, as shown in Appendix B. Thus, we prove the *tightness* of our analysis for CCH and BUH.

5 Assignment Properties

This section presents assignment properties later used in Section 6. We have two cases: when the task sets with the highest utilization in each island are given and when they are not given, that is, all cases need to be considered.

5.1 Given Highest Utilization Task Sets per Island

We define set $\mathbf{Y} = \{y_1, y_2, \ldots, y_V\}$ as a set of the indexes of task sets with the highest utilizations in each island. In other words, the set of indexes for which holds that $y_j = \ell_{j,Q}$ for all $j = 1, 2, \ldots, V$. Since the islands are ordered with respect to their voltage/frequency, we have $y_1 < y_2 < \cdots < y_V$. It is clear that task set $\mathbf{T}_M$ will always be the *highest utilization task set* in the *highest voltage/frequency island* I_V, or in other words, y_V is always equal to M. Furthermore, the *highest utilization task set* in island I_j cannot be a task set with a utilization less than $w_{j \cdot Q}$, as this would contradict the definition of set $\mathbf{Y}$. Thus, by definition, we know that $y_j \ge j \cdot Q$ for all $j = 1, 2, \ldots, V$; otherwise, there is no feasible assignment to satisfy the definition of set $\mathbf{Y}$.

The following lemma provides an important property for assigning task sets for a given set $\mathbf{Y}$.

Lemma 1: Suppose that $\frac{P(s)}{s}$ is an increasing function of s between s_{crit} and s_{max}. For a given $\mathbf{Y}$, the highest cycle utilizations of islands I_i and I_h are w_{y_i} and w_{y_h}, respectively, and it holds that $w_{y_i} \le w_{y_h}$ when $i < h$. We consider task sets $\mathbf{T}_j$ and $\mathbf{T}_k$ such that $w_j \le w_k \le w_{y_i} \le w_{y_h}$, where $\mathbf{T}_j$ is assigned to island I_i and $\mathbf{T}_k$ is assigned to island I_h. Under SFA for the given $\mathbf{Y}$, swapping the assignment such that $\mathbf{T}_j$ is assigned to island I_h and $\mathbf{T}_k$ is assigned to island I_i consumes no more energy than the original assignment.

Proof: The energy, after swapping, is *reduced* by

$$D\left[P(s_{y_h})\frac{w_j}{s_{y_h}} + P(s_{y_i})\frac{w_k}{s_{y_i}} - P(s_{y_h})\frac{w_k}{s_{y_h}} - P(s_{y_i})\frac{w_j}{s_{y_i}}\right]$$

$$= D\left[\frac{P(s_{y_h})}{s_{y_h}} - \frac{P(s_{y_i})}{s_{y_i}}\right](w_j - w_k) \le 0,$$

Algorithm 2 Greedy Algorithm for a Given $\mathbf{Y}$ Set

```
Input: The fixed set $\mathbf{Y} = \{y_1, y_2, \ldots, y_V\}$;
Output: The task set assignment with the minimal energy consumption based on the given set $\mathbf{Y}$;
1: for $j = 1$ to $V$ do
2:     $\mathbf{L}_j \leftarrow Q - 1$ task sets with utilizations less than or equal to $w_{y_j}$, in a decreasing order with respect to their utilizations;
3:     Remove the cores in set $\mathbf{L}_j$ from the problem to create a new sub-problem;
4: end for
5: return $\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_V$;
```

which concludes the lemma, since $\frac{P(s_{y_h})}{s_{y_h}} - \frac{P(s_{y_i})}{s_{y_i}} \geq 0$ and $w_j - w_k \leq 0$. □

When the *highest utilization task sets* on each island are already selected, i.e., set $\mathbf{Y} = \{y_1, y_2, \ldots, y_V\}$ is given, we can derive the optimal assignment based on Lemma 1, presented in Algorithm 2. Algorithm 2 starts by assigning $Q-1$ task sets onto cores in island I_1. Since now island I_1 has its Q cores with tasks assigned to them, the island, cores and task sets are removed from the input instance of the problem and a new sub-problem with one less element in $\mathbf{Y}$ is created. The process is repeated for islands $I_2, \ldots, I_{V-1}$ until only $Q-1$ task sets remain, which are assigned onto cores in island I_V. The overall time complexity for a given $\mathbf{Y}$ is $O(QV)$. Moreover, from Lemma 1, we have the following theorem and corollary.

Theorem 1: Suppose that $\frac{P(s)}{s}$ is an increasing function of s between s_{crit} and s_{max}. For a given $\mathbf{Y}$, Algorithm 2 derives the optimal solution for the multiple voltage islands assignment problem under SFA.

Proof: This comes directly from Lemma 1. □

Corollary 1: Under SFA and for a given set $\mathbf{Y}$, in order to assign task sets to voltage island I_j, the optimal assignment solution assigns $Q-1$ *adjacent and consecutive* task sets with the *highest utilizations* whose values are *less* than the corresponding utilization w_{y_j}, in each sub-problem in Algorithm 2.

Example: We consider $\mathbf{Y} = \{5, 7, 11, 12\}$ with $V = 4$ and $Q = 3$. The optimal assignment of the task sets onto cores belonging to different islands is shown in Figure 5. According to Algorithm 2, we first focus on island I_1, that is $j = 1$, and assign $Q-1$ task sets with utilizations less than or equal to w_{y_1} onto cores in island I_1. In this example $\mathbf{T}_5$ has the highest utilization in island I_1 ($y_1 = 5$), hence task sets $\mathbf{T}_4$ and $\mathbf{T}_3$ are also assigned onto cores belonging to I_1. Furthermore, $\mathbf{T}_3$, $\mathbf{T}_4$ and $\mathbf{T}_5$ are now removed from the sub-problem (coloured in gray). To solve the new sub-problem, we focus on island I_2, that is $j = 2$. Since $\mathbf{T}_7$ has the highest utilization on island I_2, task sets $\mathbf{T}_6$ and $\mathbf{T}_2$ are assigned onto cores belonging to I_2. These task sets are removed from the sub-problem (coloured in gray) and the process is repeated.

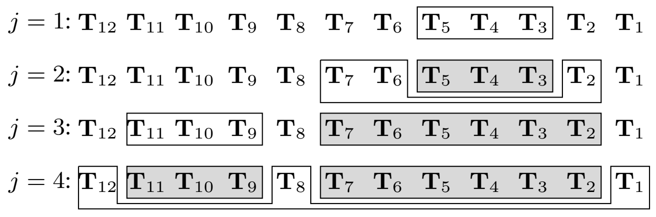

$j=1$: $\mathbf{T}_{12}$ $\mathbf{T}_{11}$ $\mathbf{T}_{10}$ $\mathbf{T}_9$ $\mathbf{T}_8$ $\mathbf{T}_7$ $\mathbf{T}_6$ $\mathbf{T}_5$ $\mathbf{T}_4$ $\mathbf{T}_3$ $\mathbf{T}_2$ $\mathbf{T}_1$
$j=2$
$j=3$
$j=4$

Fig. 5: Example for $V = 4$, $Q = 3$ and $\mathbf{Y} = \{5, 7, 11, 12\}$.

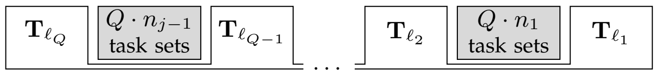


Fig. 6: Adjacent remaining task sets groups, with $n_i \in \mathbb{N}^0$.

5.2 All Possible Highest Utilization Task Sets

If the highest utilization task sets on each island are *not* given, then one possibility to derive the optimal solution is to try every possible combination for set $\mathbf{Y}$ and select the one with the minimum energy consumption. The number of combinations to be considered is at most $\binom{QV-Q}{V-1} \leq (eQ)^{V-1}$, since it holds that $\left(\frac{eN}{K}\right)^K$ is an upper bound for $\binom{N}{K}$, where e is the Euler's number. Thus, an algorithm that tries every possible combination for set $\mathbf{Y}$ and applies Algorithm 2 for each possibility, will have a time complexity $O\left(e^{V-1}Q^V V\right)$. The complexity is polynomial when V is a constant, but it is still too high and does not solve the problem in an efficient manner.

6 Dynamic Programming Solution

This section details an efficient dynamic programming to solve the *multiple voltage islands assignment problem*, called Dynamic Voltage Island Assignment (DYVIA) algorithm.

6.1 Description of the DYVIA Algorithm

The algorithm is based on a property described in Theorem 2, which comes as a direct result of Corollary 1. Such property is illustrated in Figure 6.

We define set $\mathbf{L} = \{\ell_0, \ell_1, \ell_2, \ldots, \ell_Q\}$ as the indexes of the task sets assigned onto cores in one *general* island (as opposed to set $\mathbf{L}_j$, defined for the particular island I_j) such that $\ell_1 < \ell_2 < \cdots < \ell_Q$, with ℓ_0 auxiliary and less than ℓ_1. Naturally, it holds that $w_{\ell_1} \leq w_{\ell_2} \leq \cdots \leq w_{\ell_Q}$.

Theorem 2: When assigning task sets with indexes $\{\ell_1, \ell_2, \ldots, \ell_Q\}$ onto cores in island I_j, in the optimal solution under SFA, all the task sets assigned onto cores in islands $I_1, I_2, \ldots, I_{j-1}$ will form up to $j-1$ groups of *remaining adjacent task sets* and every group will have a number of task sets that is a multiple of Q.

Proof: From Theorem 1 and Corollary 1, it is quite clear that Theorem 2 holds for any given $\mathbf{Y}$ set, including the optimal one. □

For given (i, j), let set $\Lambda(i, j)$ be the set of all possible $\mathbf{L}$ sets that satisfy Theorem 2. That is, $\Lambda(i, j)$ stores the *potentially optimal combinations*, such that $\ell_0 = i-1$, $\ell_Q = j$

and $\ell_h = \ell_{h-1} + 1 + nQ$ for $0 < h < Q$ with $\ell_h < j$ and $n \in \mathbb{N}^0$.

We define $G(i,j)$ as our *dynamic programming* function, where i is the index number of the first task set and j the index number of the last task set to be considered in this sub-problem. Function $G(i,j)$ will return the minimum energy consumption for the assignment of task sets $\mathbf{T}_i, \mathbf{T}_{i+1}, \ldots, \mathbf{T}_{j-1}, \mathbf{T}_j$ onto cores, using a number $v = \frac{j-i+1}{Q}$ of islands (from Corollary 1, $j-i+1$ will always be an integer multiple of Q). Particularly, function $G(i,j)$ only focuses on the *highest voltage/frequency island* in the sub-problem and the task sets $\mathbf{T}_{\ell_1}, \mathbf{T}_{\ell_2}, \ldots, \mathbf{T}_{\ell_Q}$ that are to be assigned onto cores in this island. To derive the optimal assignment of task sets onto cores that resulted in this minimal energy consumption, a standard backtracking technique may be used, e.g., building an additional table $BTG(i,j)$ with the task sets indexes $\{\ell_1, \ell_2, \ldots, \ell_Q\}$ that result in the minimum energy consumption for sub-problem $G(i,j)$.

From the definition, just as any other island, the *highest voltage/frequency island* on each $G(i,j)$ sub-problem will hold Q cores. It is easy to see that task set $\mathbf{T}_j$ is always assigned onto a core in this *highest voltage/frequency island* and that, according to SFA, its cycle utilization defines the frequency of the cores and the voltage of the island. Therefore, function $G(i,j)$ has to decide which of the remaining $Q-1$ task sets are assigned to the *highest voltage/frequency island* on the sub-problem. In order to do this, all the *potentially optimal combinations* that satisfy Theorem 2, i.e., $\Lambda(i,j)$, need to be considered.

The energy consumption of the *highest voltage/frequency island* for each combination is computed through function $H(\mathbf{L})$, which is similar to Equation (1), but for set $\mathbf{L}$ instead of set $\mathbf{L}_j$. Function $H(\mathbf{L})$ is defined as

$$H(\mathbf{L}) = \begin{cases} 0 & \text{if } \sum_{i=1}^{Q} w_{\ell_i} = 0 \\ D\left(\eta + \frac{P(s_{\ell_Q})}{s_{\ell_Q}} \sum\limits_{i=1}^{Q} w_{\ell_i}\right) & \text{otherwise,} \end{cases} \tag{5}$$

where $s_{\ell_Q} = \max\{s_{\text{crit}}, w_{\ell_Q}\}$.

The total energy consumption for each combination comes from the summation of $H(\mathbf{L})$ and the minimum energy consumption of each sub-problem for every group of remaining adjacent task sets, as stated in Theorem 2. Once the total energy consumption for every combination is obtained, the minimum one is chosen as the result of the problem.

The initial conditions for building the dynamic programming table are defined in Equation (6), as

$$G(i,j) = \begin{cases} 0 & \begin{array}{l}\text{if } i = 0 \text{ or} \\ \sum_{h=i}^{j} w_h = 0 \text{ or} \\ 0 \le j < i+Q-1 \le QV\end{array} \\ D\left(\eta + \frac{P(s_j)}{s_j} \sum\limits_{h=i}^{j} w_h\right) & \text{otherwise,} \end{cases} \tag{6}$$

where $s_j = \max\{s_{\text{crit}}, w_j\}$. Equation (6) builds the table for all sub-problems with Q consecutive task sets, i.e., $j-i+1 = Q$.

Algorithm 3 DYVIA

Input: Number of voltage islands V and cores per island Q;
Output: The minimal energy consumption under SFA;

```
    {Initialize the table according to Equation (6)}
 1: for $h = 1$ to $Q(V-1)$ do
 2:     $s_j \leftarrow \max\{s_{\text{crit}}, w_{h+Q-1}\}$;
 3:     $G(h, h+Q-1) \leftarrow \eta D + P(s_j) \cdot \frac{D}{s_j} \sum_{n=h}^{h+Q-1} w_n$;
 4: end for
    {Fill the table (Bottom-Up) according to Equation (7)}
 5: for $k = 2$ to $V-1$ do
 6:     for $h = 1$ to $Q(V-k)$ do
 7:         $G(h, h+kQ-1) \leftarrow \infty$;
 8:         Obtain $\Lambda(h, h+kQ-1)$;
 9:         for all $\mathbf{L} \in \Lambda(h, h+kQ-1)$ do
10:             $E \leftarrow H(\mathbf{L})$;
11:             for $n = 1$ to $Q$ do
12:                 $E \leftarrow E + G(\ell_{n-1}+1, \ell_n - 1)$;
13:             end for
14:             if $E < G(h, h+kQ-1)$ then
15:                 $G(h, h+kQ-1) \leftarrow E$;
16:             end if
17:         end for
18:     end for
19: end for
20: $G(1, QV) \leftarrow \infty$;
21: Obtain $\Lambda(1, QV)$;
22: for all $\mathbf{L} \in \Lambda(1, QV)$ do
23:     $E \leftarrow H(\mathbf{L})$;
24:     for $n = 1$ to $Q$ do
25:         $E \leftarrow E + G(\ell_{n-1}+1, \ell_n - 1)$;
26:     end for
27:     if $E < G(1, QV)$ then
28:         $G(1, QV) \leftarrow E$;
29:     end if
30: end for
31: return $G(1, QV)$;
```

The recursive function is defined in Equation (7), as

$$G(i,j) = \min_{\forall \mathbf{L} \in \Lambda(i,j)} \left\{ H(\mathbf{L}) + \sum_{n=1}^{Q} G(\ell_{n-1}+1, \ell_n - 1) \right\}. \tag{7}$$

The bottom-up implementation of the dynamic programming solution is presented in Algorithm 3.

Example: We consider a system with $V = 3$ and $Q = 3$ to conceptually illustrate how algorithm DYVIA works. The solution to the problem, i.e., the optimal energy consumption under SFA, is found in entry $G(1,9)$ of our dynamic programming table. In order to build entry $G(1,9)$ according to Equation (7), the algorithm checks all $\mathbf{L} \in \Lambda(1,9)$, i.e., all *potentially optimal combinations* (combinations satisfying Theorem 2). Figure 7 shows these combinations, where for each case, the task sets assigned to I_3 are boxed in white and the resulting sub-problems are coloured in gray. As stated in Theorem 2, every sub-problem contains 3 or 6 task sets. For each combination, function $H(\mathbf{L})$ computes the energy consumption for island I_3 and the algorithm refers to the entries built previously to obtain the lowest

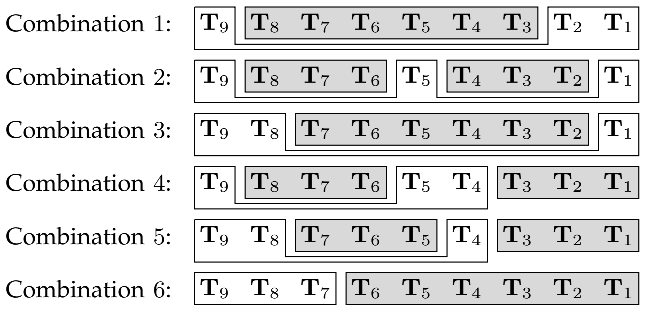


Fig. 7: Example of algorithm DYVIA for building $G(1,9)$, with $V=3$ and $Q=3$.

energy consumption under SFA of the resulting sub-problems. Sub-problems $G(1,3)$, $G(2,4)$, …, $G(7,9)$ are built as initial conditions, according to Equation (6). Sub-problems $G(1,6)$, $G(2,7)$ and $G(3,8)$ are solved according to Equation (7), just as $G(1,9)$.

For the backtracking, we simply store in another table the indexes of the combination that results in the minimum energy consumption for each sub-problem. For example, in case that *Combination* 3 derives the best result for $G(1,9)$ in Figure 7, then we would store $\{1,8,9\}$ in entry $BTG(1,9)$. Once the algorithm finishes, we look into $BTG(1,9)$ for the task sets that should be assigned to I_3. Then, we look into $BTG(2,7)$ to know which task sets should be assigned to I_2, and so forth.

Due to Theorem 2, for this example, DYVIA only checks among 6 potentially optimal combinations *for* $G(1,9)$, *instead of the* 28 brute force *combinations for all possibilities.*

6.2 Complexity Analysis for DYVIA

We now analyse the time complexity to build table G in Algorithm 3. To build $G(1,M)$, all the possible combinations that satisfy Theorem 2 will need to be considered for this sub-problem. This can be considered equivalent to choosing $V-1$ groups of Q task sets from a set of $\frac{(V-1)Q}{Q}+Q-1$ task sets, which results in $\binom{Q+V-2}{V-1}$ combinations. For the resulting iQ sub-problems with $(V-i)Q$ task sets for $i=1,2,\ldots,V-1$, similarly, the number of combinations is $\binom{Q+V-2-i}{V-1-i}$ for $i=1,2,\ldots,V-1$. Therefore, the total amount of iterations needed to build the table, denoted as Z, is

$$Z=\binom{Q+V-2}{V-1}+Q\sum_{i=1}^{V-1}i\binom{Q+V-2-i}{V-1-i}$$

or

$$Z=\binom{Q+V-2}{Q-1}+Q\sum_{i=1}^{V-1}(V-i)\binom{Q-2+i}{Q-1}.$$

Which by applying the Hockeystick Identity and Pascal's rule, can be rephrased as

$$Z=\binom{Q+V-2}{Q-1}+Q\binom{Q+V-1}{Q+1}$$

or

$$Z=\binom{Q+V-2}{V-1}+Q\binom{Q+V-1}{V-2}.$$

Furthermore, since $\left(\frac{eN}{K}\right)^K$ is an upper bound for $\binom{N}{K}$, we have

$$Z\le\left(\frac{e\,(Q+V-2)}{Q-1}\right)^{Q-1}+Q\left(\frac{e\,(Q+V-1)}{Q+1}\right)^{Q+1}$$

or

$$Z\le\left(\frac{e\,(Q+V-2)}{V-1}\right)^{V-1}+Q\left(\frac{e\,(Q+V-1)}{V-2}\right)^{V-2}.$$

Therefore, $min\left\{O\left(\frac{(eV)^{Q+1}}{Q^Q}\right),O\left(\left(\frac{eQ}{V}\right)^{V-1}\right)\right\}$ is the time complexity of DYVIA, which is more efficient than the greedy algorithm presented in Section 5.2 and polynomial when either V or Q is a constant.

6.3 Optimal Mapping Under SFA vs. Optimal DVFS

Even though DYVIA is optimal with respect to energy consumption when using SFA for individual islands, there may exist better assignments when each individual island is free to use any DVFS algorithm, e.g., [18]. We now analyse the approximation factor of DYVIA, against the optimal DVFS algorithms. Suppose that AF_{SFA} is the approximation factor (by definition, the worst case) of SFA [15], against the ideally optimal DVFS scheduling for the task sets assigned on *a single voltage island*.

We define different total energy consumptions as $E^{\text{RUN=SFA}}_{\text{ASG=DYVIA}}$, $E^{\text{RUN=DVFS}}_{\text{ASG=DYVIA}}$, $E^{\text{RUN=SFA}}_{\text{ASG=DVFS}}$, and $E^{\text{RUN=DVFS}}_{\text{ASG=DVFS}}$, where RUN=SFA means that SFA is used for executing on individual islands, RUN=DVFS means that an optimal DVFS algorithm is used for executing on individual islands, ASG=DYVIA means that task sets are assigned to islands using DYVIA, and ASG=DVFS means that task sets are assigned to islands using an algorithm which is optimal when using an optimal DVFS algorithm for executing on individual islands. Clearly, the optimal energy consumption is $E^{\text{RUN=DVFS}}_{\text{ASG=DVFS}}$.

We know that $E^{\text{RUN=DVFS}}_{\text{ASG=DYVIA}}\le E^{\text{RUN=SFA}}_{\text{ASG=DYVIA}}$ by the definition that the optimal DVFS strategy is adopted. Additionally, we know that $E^{\text{RUN=SFA}}_{\text{ASG=DYVIA}}\le E^{\text{RUN=SFA}}_{\text{ASG=DVFS}}$, since we know that DYVIA is optimal if individual islands use SFA and any other voltage island assignment will consume the same or more energy. Moreover, according to the definition of the approximation factor for the energy consumption of SFA in a single voltage island, we have $E^{\text{RUN=SFA}}_{\text{ASG=DVFS}}\le \text{AF}_{\text{SFA}}\cdot E^{\text{RUN=DVFS}}_{\text{ASG=DVFS}}$.

Finally, the approximation factor of the total energy consumption for DYVIA is presented in Equation (8).

$$E^{\text{RUN=SFA}}_{\text{ASG=DYVIA}}\le \text{AF}_{\text{SFA}}\cdot E^{\text{RUN=DVFS}}_{\text{ASG=DVFS}} \tag{8}$$

Therefore, the approximation factor of DYVIA against the optimal solutions with DVFS for energy minimization is the same as the approximation factor of SFA on a single voltage island. It has been shown in [15] that the

Frequency [MHz]	Minimum Voltage [Volts]	Idle Power [Watts]	Execution Power [Watts]
100	0.8	19.237213	22.441350
106	0.8	19.402374	22.538383
114	0.8	19.402374	22.867801
123	0.8	19.468439	23.165997
133	0.8	19.468439	23.560321
145	0.8	19.763666	24.119624
160	0.8	19.794633	24.584324
178	0.8	19.864827	25.348004
200	0.8	20.129086	26.168786
228	0.8	20.391280	27.293623
266	0.8	20.756701	28.771025
320	0.8	21.280769	30.674611
400	0.8	21.811671	33.449853
533	0.8	23.132058	38.052265
800	1.1	44.549986	84.395704

TABLE 1: Experimental Power Profile for SCC.

approximation factor of SFA against the optimal DVFS is dependent on the power consumption model and the number of cores per voltage island. For practical cases, if $P(s) = \alpha s^\gamma + \beta$, using SFA for task execution has an approximation factor of at most 1.42 (1.53, 1.63, 1.74, respectively), when the voltage island has up to 2 (4, 6, 8, respectively) cores.

7 Experimental Evaluation on SCC

This section presents experimental evaluations conducted on SCC. We compare the energy consumption and execution time of CCH, BUH, EOH (by taking CCH as the initial solution and with 200 iterations for seeking improvement of the current solution), and DYVIA.

7.1 Experimental Setup

The experiments are conducted on Intel's Single-Chip Cloud Computer (SCC), a research platform that integrates 48 cores on a single chip, where the individual IA P54C cores run at 100-800 MHz. Intel's SCC runs a single-core Linux (kernel version 3.1.4) on each core.

Algorithms EOH and DYVIA need a power consumption profile of the system, since both algorithms compute the expected energy consumption to compare different assignments. Table 1 presents the results of experimental measurements conducted on SCC to obtain such power consumption profile. The table shows all the available execution frequencies, their corresponding minimum island voltages for stable execution[2], together with the measured idle[3] and execution power consumptions for all 48 cores. Some error is present in this power profile. Part of it is due to the resolution of the voltage and current meter inside SCC (around 0.3 Watts resolution), but mostly because our SCC platform has one *faulted* core, and it is not possible to estimate the power consumption of this core for each frequency. Furthermore, Table 1 shows that if a core configured to execute at a high frequency is idle, it would consume more energy than idling a core configured to execute at a low frequency. However, the 18.298 Watts of idle power consumption for always having all voltage islands active and all cores idle at 100 MHz is an offset that no algorithm can improve, because it is not possible to shut down cores or voltage islands on SCC. Therefore, for this power profile, s_{crit} and η are both considered zero in Equations (1), (6) and (7). Moreover, in the experiments, this 18.298 Watts offset is subtracted from the measurements to correctly compare the performance of the different algorithms.

For the tasks, we consider two different *single core* benchmarks: (1) an FFT digital filter and (2) an edge detection algorithm for images. Each instance of a benchmark represents one task, and each task is periodically executed in the assigned core. For every task instance of the FFT benchmark, the input for the FFT filter is a discrete signal consisting of $100,000$ samples, at 100 kHz sampling frequency, of two sine waves added together (5 kHz and 12 kHz) plus different random noise for each run. For tasks using the edge detection algorithm, the input is a 640x480 pixels bmp image, randomly chosen for each period, among a database of $2,500$ pictures. That is, for all tasks using one type of benchmark, the input is randomly chosen for each period, but the size of the input remains constant. This is done such that we can obtain a lower-bound for the worst-case execution cycles of both benchmarks through experimental measurements on the SCC. Specifically, we execute both benchmarks $1,000$ times at every available frequency on SCC, and we use the highest measured value for each benchmark as lower-bound for the worst-case execution cycles for each frequency (the detailed measurements are presented in Appendix C). This way, we consider the effects of cache and memory access, whose access time is not scaled when the frequency of a core changes. Namely, this means that the worst-case execution cycles for all task instances of one benchmark type is the same for all tasks executing at the same frequency. To obtain different cycle utilization for each task, the period in which the benchmarks are executed is set accordingly. For example, according to our experiments the FFT benchmark has a lower-bound for the worst-case execution cycles at 800 MHz of $1.92 \cdot 10^9$ cycles. Thus, if task τ_j has a desired cycle utilization of 10^9 cycles per second, we set the period of the task to $p_j = \frac{c_j}{w_j} = 1.92$ seconds.

Since our SCC platform has one *faulted* core, there are only 47 available cores in the platform for our experiments. In order to evaluate the performance of the algorithms for different task sets, 12 arbitrary cases are considered. Every one of the 12 cases consists of 200 tasks with different cycle utilization, 100 of each benchmark type. The tasks are partitioned using LTF with $M' = 47$

2. In *The SCC Programmer's Guide* v1.0, the minimum voltage for stable execution bellow 460 MHz is 0.7 Volts. However, in RCCE v2.0 this was changed to 0.8 Volts due to stability problems.

3. The changes in the idle power for different frequencies, observed in Table 1, are due to background processes of the operating system.

(all the cores available on *our* SCC). The benchmarks are *single core* applications, thus, no communication between cores is needed after the task sets are assigned onto cores.

We have integrated all the mapping algorithms as a software written in C++ that runs on a single core. Each algorithm decides the assignment, configures the voltage of the islands and the frequencies of the cores accordingly, and finally executes the benchmarks on the corresponding cores. Since the purpose of this experiment is to evaluate the performance of the resulting assignment in terms of energy consumption, we measure the total energy of the SCC chip for 100 seconds after executing the benchmarks.

Furthermore, we conduct a separate experiment to measure the average execution time of each algorithm on SCC at 533 MHz. To see the effects of V and Q in the execution time of each algorithm, the average execution time experiments are conducted for twelve hypothetical system configurations with different V and Q values, i.e., $V = \{2, 4, 6\}$ and $Q = \{2, 4, 6, 8\}$. All four algorithms need to configure the voltages of the islands, the frequencies of the cores, and assign the task sets onto cores. Hence, we only measure the time of the mapping decision process. Each algorithm is executed $10,000$ times and the average execution time is presented.

7.2 Experimental Results

Table 2 presents the experimental measurements of energy consumption on SCC (the average value among 10 consecutive executions during 100 seconds each) for the 12 cases of different benchmark utilizations, compared against the *expected energy consumption*. The *expected energy consumption* is computed by running the algorithms, considering the cycle utilization of each task set and the same power profile used by EOH and DYVIA to estimate the energy consumption, i.e., Table 1. The experimental energy consumption values from Table 2 are obtained through integration of power measurements, i.e., every 1 millisecond the power consumption of all cores is measured, this power is multiplied by the elapsed time from the previous measurement, and added to the total measured energy consumption.

The measured energy values in Table 2 have, in average, a 3.62% error from the *expected energy consumption*. This is due to the resolution of the voltage and current meter inside SCC, the presence of one *faulted* core in our SCC platform, the actual execution cycles of each task instance (which could be different from the expected worst-case execution cycles), and the intrinsic integration error from the 1 millisecond resolution in which we measure the total power when computing the consumed energy[4]. Naturally, since DYVIA is optimal when using SFA in individual islands, the *expected energy consumption* of DYVIA is always the lowest. However for configurations where the heuristics provide good results, the experimental measurements may show values where DYVIA has higher energy consumption, but this effect is only due to the measurement errors mentioned before.

Furthermore, we compute the ratios between the energy consumption of the heuristics and the energy consumption of DYVIA, for each one of the 12 cases from Table 2 (expected and measured). Thus, Table 3 presents the minimum, average and maximum ratios for each heuristic, among these 12 cases. We can observe that the measured energy ratios are quite similar to the expected energy ratios. Specifically, the measured average and maximum energy ratios are 1.0579 and 1.1297 for CCH, respectively; 1.1048 and 1.3464 for BUH, respectively; and 1.0324 and 1.0986 for EOH, respectively.

Finally, Figure 8 presents the experimental results of the average execution time of each algorithm on SCC, for the twelve hypothetical system configurations with different V and Q values. As expected, heuristics CCH and BUH complete their executions very quickly due to their low complexity. As for EOH and DYVIA, the average execution time increases when Q or V increases, and is bounded by 226 ms, which is still suitable for *online* usage. Moreover, DYVIA is much faster than EOH in all the evaluated cases, except when $V = 6$ and $Q = 8$, where EOH and DYVIA have similar execution times.

12 cases	Expected Energy Consumption [Joule]				Measured Energy Consumption [Joule]			
	CCH	BUH	EOH	DYVIA	CCH	BUH	EOH	DYVIA
1	1066.3	1066.3	1066.3	1004.5	1072.4	1066.3	1067.4	996.1
2	1085.6	1085.6	1085.6	1024.8	1050.3	1073.1	1072.9	1009.3
3	2144.8	2572.3	2144.8	2045.1	2068.9	2425.9	2057.3	1913.4
4	1070.9	1070.9	1070.9	1013.6	993.9	1029.8	990.5	961.9
5	773.4	773.1	773.4	766.4	787.4	798.5	771.9	811.8
6	2181.9	2620.0	2181.9	2093.0	2090.6	2373.6	2085.8	1981.5
7	952.5	931.2	931.6	895.6	943.1	919.0	933.5	882.9
8	950.8	940.1	920.8	894.3	941.3	916.9	922.5	888.6
9	2120.4	2120.4	2026.0	2013.8	2091.8	2074.5	1896.5	1974.7
10	2081.9	2557.3	2081.9	2006.1	2001.8	2450.4	1999.4	1820.0
11	854.0	854.0	797.6	777.5	744.6	751.5	713.4	728.6
12	890.4	890.4	798.0	797.9	904.7	883.1	810.3	800.8

TABLE 2: Experimental Evaluation Results on SCC.

Algorithm	Expected Energy Ratio			Measured Energy Ratio		
	Min.	Avg.	Max.	Min.	Avg.	Max.
CCH	1.0092	1.0591	1.1159	0.9700	1.0579	1.1297
BUH	1.0088	1.1107	1.2748	0.9837	1.1048	1.3464
EOH	1.0002	1.0348	1.0615	0.9509	1.0324	1.0986

TABLE 3: Experimental Energy Ratios Results on SCC.

8 Hypothetical Platforms Simulations

This section presents simulations for hypothetical platforms with different V and Q values, to analyse the performance of CCH, BUH, EOH and DYVIA for energy minimization.

4. Additional details are included in Appendix E.

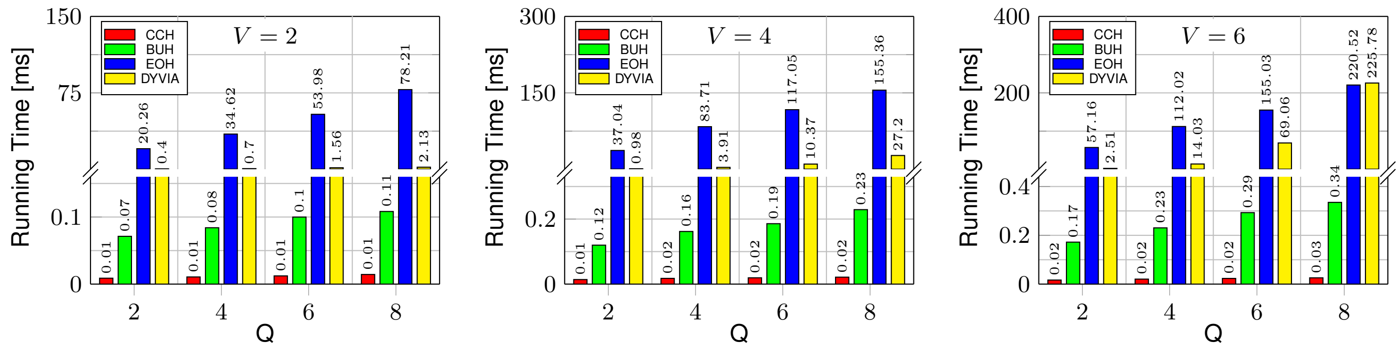


Fig. 8: Experimental results on SCC (at 533 MHz) of average execution time for $V = 2$, $V = 4$ and $V = 6$.

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

(a) Frequency vs. voltage

Voltage [Volts]	Total Power [Watts]
0.70	25.38
0.80	37.26
0.91	50.76
1.00	70.73
1.05	91.25
1.10	110.15
1.14	125.27
1.21	161.99
1.28	201.40

(b) Power vs. voltage

TABLE 4: Experimental Measurement from [8] (48-cores).

Frequency [MHz]	Execution Power [Watts]	Energy for executing 10^8 computer cycles simultaneously in all cores [Joule]
242.7	25.38	10.46
464.5	37.26	8.02
686.7	50.76	7.39
851.6	70.73	8.31
936.6	91.25	9.74
1016.9	110.15	10.83
1077.8	125.27	11.62
1177.0	161.99	13.76
1267.0	201.40	15.90

TABLE 5: Power Profile from Measurements in [8].

8.1 Simulation Setup

To see the effects of V and Q in the performance of the algorithms (with respect to energy), the simulations are conducted for the same twelve hypothetical system configurations as in the average execution time experiments from Section 7, i.e., $V = \{2, 4, 6\}$ and $Q = \{2, 4, 6, 8\}$.

We would also like to test the algorithms in systems with richer DVFS and DPM features than SCC. For such a purpose, we consider the experimental results from [8], in which a multi-core system that integrates 48 cores was developed. For completeness, Figure 12 and Figure 13 from [8] are summarized in Table 4. Table 4a relates several frequencies for the cores and their minimum voltages for stable execution. Table 4b shows power consumption values for running all cores at certain voltages (and at their corresponding maximum frequencies for each voltage). We approximate the results in Table 4a by using a quadratic function, and then use such a function and the values from Table 4b to relate frequency with power. This derives a new power profile[5] (for all 48 cores running together at the same voltage/frequency), which is presented in Table 5. We use the frequencies shown in Table 5 as the available execution frequencies for the cores, and divide the total execution power by 48 to obtain the power consumed by each individual core. Moreover, Table 5 also shows the energy consumption for executing 10^8 computer cycles (simultaneously in all 48 cores) at each corresponding frequency. Clearly, 686.7 MHz is the *critical frequency* for such a power profile. Hence, for this power profile, we never execute at a frequency bellow 686.7 MHz, because even though it might still meet the timing constraints, doing so would consume unnecessary energy. Finally, since the work in [8] makes no reference to the power consumed by an island for been active, we use $\eta = 0$.

Regarding DPM, we consider that the cores can be put to a low-power mode when they have no workload to execute. When a core is in such a low-power mode, we consider that it consumes β' power. Similar to the experimental measurements from Section 7, $\beta' M$ (the power consumption for having all the cores in the low-power mode) is an offset that no algorithm can improve. Therefore, in the simulations we set $\beta' = 0$, to correctly compare the performance of the different algorithms.

For each hardware configuration (combination of V and Q), we consider 100 different random cases of synthetic tasks. For each case, the total amount of tasks, denoted as N, is randomly chosen between M and $10M$. The cycle utilizations and periods of the tasks are also

5. The frequency and power consumption values in Table 5, for each one of the 48 cores, can be modelled, e.g., with power consumption function $P(s) = \alpha s^{\gamma} + \beta$, where $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\gamma = 3$, and $\beta = 0.5$ Watts. 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.

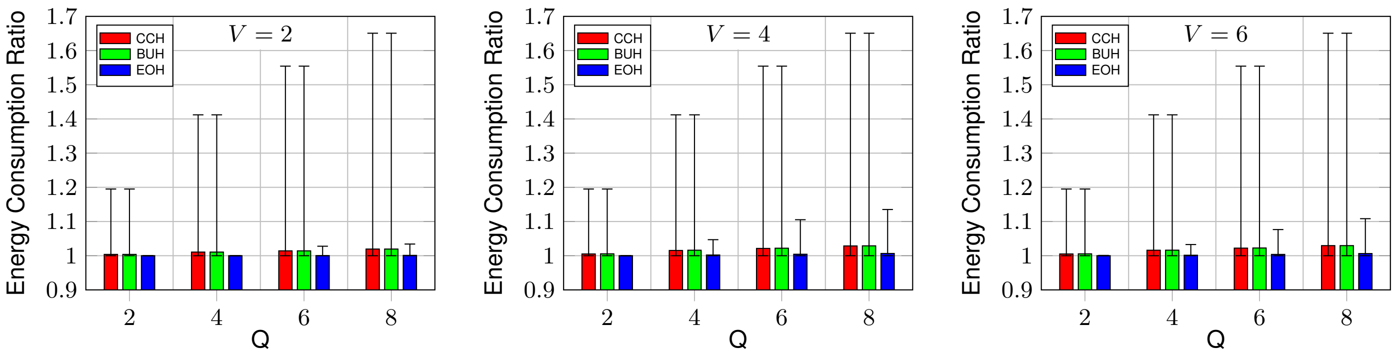


Fig. 9: Simulation results of energy consumption for $V = \{2, 4, 6\}$, where the range of the energy consumption ratios of a configuration is shown by the vertical line and the bar represents the average energy consumption ratio. All four algorithms partition tasks with LTF by using all the available cores, i.e., $M' = M$ for each configuration.

randomly chosen. The tasks are partitioned using the *Largest-Task-First* (LTF) strategy described in Section 3. We consider two different policies to choose the value of M' for LTF. For the first policy, we partition the N tasks into $M' = M$ task sets with LTF, such that all the available cores are utilised. For the second policy, we partition the N tasks into $M' = 1, 2, \ldots, M$ task sets with LTF, then execute the corresponding *voltage island assignment algorithm* for each different resulting partition (considering $M - M'$ dummy task sets), and finally choose the partition that results in the lowest overall energy consumption for the corresponding algorithm. Clearly, the second policy incurs higher time complexity, but can potentially save more energy.

8.2 Simulation Results

Figure 9 presents the simulation results for energy consumption when all the four algorithms consider only one task partition obtained with LTF by using all the available cores, i.e., $M' = M$. The simulation results are shown as ratios between the energy consumption of the heuristics and the energy consumption of DYVIA, for the different hypothetical system configurations. Since for each configuration, 100 different cases are considered, the maximum and minimum energy consumption ratios (among the 100 cases) are shown by a vertical line and the bar represents the average energy consumption ratio.

Since DYVIA is optimal when using SFA in individual islands, the simulation results show that the minimum energy consumption ratio is never lower than 1, since the heuristics never consume less energy than DYVIA. In the average cases, we observe that all heuristics behave rather well, since the average energy consumption ratio for all heuristics is at most 1.0295. As for the maximum energy consumption ratio, the simulation results show that it increases with the value of Q. This is an expected effect, since lower Q values imply that the system is closer to the ideal per-core DVFS, and simple heuristics have higher chances of providing reasonable solutions. Specifically, the maximum ratio can go up to 1.6507 for CCH and BUH, and up to 1.1351 for EOH.

For all the evaluated cases, considering different amounts of cores for the task partitioning stage and then choosing the partition that derives the lowest energy consumption, resulted in insignificant improvements, even at the cost of higher time complexity. The figures (for example where all algorithms iterate through $M' = 1, 2, \ldots, M$ possible partitions) are therefore omitted, because they have no visible difference with Figure 9.

9 Concluding Remarks

In this paper we present the analysis and solutions for the multiple voltage island assignment problem, by using SFA for individual islands. We analyse the worst-case performance for the energy consumption of several simple heuristics. Based on dynamic programming, we develop algorithm DYVIA, which derives optimal solutions for any task partition under SFA. Moreover, we also provide analysis to show that the approximation factor of DYVIA with SFA is the same as the approximation factor of SFA in a single island. We evaluate the performance and average running time of the algorithms on Intel's SCC. In all the evaluated cases, DYVIA derives solutions with the minimum energy consumption under SFA, by taking on average no longer than 225 ms to execute. However, due to the limited available supply voltages for stable executions in SCC, as shown in Table 1, and the incapability for DPM to put cores in low-power modes, in the experiments the heuristics consume at most 29% more energy than DYVIA.

Additionally, we conduct further simulations for hypothetical platforms with different combinations for the number of voltage islands and the number of cores per island, as well as richer DVFS and DPM features than SCC. We also consider different policies for the task partitioning with LTF. The simulation results show that for such platforms, the heuristics behave well in the average cases. However for the worst cases, the heuristics can consume up to 1.65 times the energy of

DYVIA, and this value increases when there are more number of cores per island.

We prove that the time complexity of DYVIA is exponential in V or Q, and polynomial if either value is a constant. Since DYVIA is optimal under SFA, it is the best choice to map the task sets, as long as its execution time is within tolerable limits. For systems with many cores per island, the simple heuristic CCH can be adopted, given that it provides reasonable solutions in terms of energy consumption for general cases, with extremely low (linear-time) complexity.

Acknowledgements

This work was partly supported by Baden Württemberg MWK Juniorprofessoren-Programms, a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 117913], and a grant from the Germany / Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the DAAD - German Academic Exchange Service (Reference No. [G_HK016/11]).

References

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

[2] S. Borkar, "Thousand core chips: a technology perspective," in *the 44th Design Automation Conference (DAC)*, 2007, pp. 746–749.

[3] J.-J. Chen and L. Thiele, "Energy-efficient scheduling on homogeneous multiprocessor platforms," in *SAC*, 2010, pp. 542–549.

[4] P. Choudhary and D. Marculescu, "Hardware based frequency / voltage control of voltage frequency island systems," in *CODES + ISSS*, 2006, pp. 34–39.

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

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

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

[8] 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," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, 2011.

[9] Intel Corporation, "Single-chip cloud computer (SCC)." [Online]. Available: http://www.intel.com/content/www/us/en/research/intel-labs-single-chip-cloud-overview-paper.html

[10] W. Jang, D. Ding, and D. Z. Pan, "A voltage-frequency island aware energy optimization framework for networks-on-chip," in *ICCAD*, 2008, pp. 264–269.

[11] H. Kim and others, "Total energy minimization of real-time tasks in an on-chip multiprocessor using dynamic voltage scaling efficiency metric," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 27, no. 11, pp. 2088–2092, 2008.

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

[13] N. Nikitin and J. Cortadella, "Static task mapping for tiled chip multiprocessors with multiple voltage islands," in *Conference on Architecture of Computing Systems (ARCS)*, 2012, pp. 50–62.

[14] Ü. Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-frequency island partitioning for gals-based networks-on-chip," in *DAC*, 2007, pp. 110–115.

[15] S. Pagani and J.-J. Chen, "Energy efficiency analysis for the single frequency approximation (SFA) scheme," in *RTCSA*, 2013, pp. 82–91.

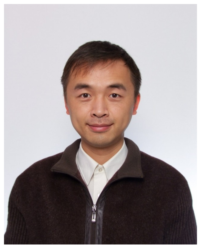
[16] E. Seo, J. Jeong, S.-Y. Park, and J. Lee, "Energy efficient scheduling of real-time tasks on multicore processors," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1540–1552, 2008.

[17] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé, "Energy-efficient policies for embedded clusters," in *Conf. on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, 2005, pp. 1–10.

[18] 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.

Santiago Pagani is a Ph.D. student and part of the research staff at the Department of Informatics in Karlsruhe Institute of Technology (KIT) in Germany. He received his Diploma in Electronics Engineering from the Department of Electronics, National Technological University (UTN), Argentina in 2010. From 2003 until 2012, he worked as a hardware and software developer in the industry sector for several companies in Argentina. He joined KIT and started his doctoral research in March 2012. His research interests include real-time systems, embedded systems, energy-efficient scheduling, power-aware designs and temperature-aware scheduling. He received a Best Paper Award from IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) in 2013.

Jian-Jia Chen is a Professor in the Department of Informatics in TU Dortmund University in Germany. He was a Juniorprofessor in the Department of Informatics in Karlsruhe Institute of Technology (KIT) in Germany from May 2010 to March 2014. He received his Ph.D. degree from Department of Computer Science and Information Engineering, National Taiwan University, Taiwan in 2006. He received his B.S. degree from the Department of Chemistry at National Taiwan University 2001. Between Jan. 2008 and April 2010, he was a postdoc researcher at Computer Engineering and Networks Laboratory (TIK) in ETH Zurich, Switzerland. His research interests include real-time systems, embedded systems, energy-efficient scheduling, power-aware designs, temperature-aware scheduling, and distributed computing. He received Best Paper Awards from ACM Symposium on Applied Computing (SAC) in 2009 and IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) in 2005 and 2013.



Minming Li received the BE and PhD degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2002 and 2006, respectively. He is currently an assistant professor in the Department of Computer Science, City University of Hong Kong. His research interests include wireless ad hoc networks, algorithm design and analysis, and combinatorial optimization.

APPENDIX A
SUMMARY OF SYMBOLS

For simple reference, Table 6 summarizes all the symbols used in the paper.

APPENDIX B
APPROXIMATION FACTOR OF BUH: EXAMPLE

This appendix shows how to extend the example from Section 4.3 for any value of Q. Thus, we prove the *tightness* of our analysis for CCH and BUH for all Q. Consider a system with $V = 16$ voltage islands, $Q = 16$ cores per island, $\eta = 0$ island power consumption, $P(s) = 2\frac{\text{Watts}}{\text{GHz}^3} \cdot s^3$ power consumption per core, and a hyper-period among all tasks of 1 second.

By solving $2(Q-1)x_{\max}^3 + 3x_{\max}^2 - 1 = 0$ when Q is 16, we know that for this value of Q, Equation (4) is maximized when $x = x_{\max} = 0.2917$. That is, if after the task partitioning stage, the task set with highest cycle utilization is $w_{256} = 10^9$, then, the worst-case happens if we have $Q-1$ task sets with cycle utilization $2.917 \cdot 10^8$, and $(Q-1)V$ task sets with cycle utilization zero. That is, $w_{256} = 10^9, w_{255} = w_{254} = \cdots = w_{241} = 2.917 \cdot 10^8, w_{240} = w_{239} = \cdots = w_1 = 0$.

For this example, both CCH and BUH will assign task sets $\mathbf{T}_{241}, \mathbf{T}_{242}, \ldots, \mathbf{T}_{256}$ to island I_{16}, which according to Equation (1) results in an energy consumption under SFA of 10.75 Joule. The optimal solution however, will assign one task set per island. That is, task set $\mathbf{T}_{241}$ to island I_1, $\mathbf{T}_{242}$ to island I_2, ..., and $\mathbf{T}_{256}$ to island I_{16}, which results in an energy consumption of 2.74 Joule. *Since we have only one task set assigned to each voltage island, this is the same energy consumption that we would obtain by having a per-core DVFS platform*. The ratio between these two energy consumptions is 3.92, which corresponds to Equation (4) and Figure 4 when Q is 16, since for such a case we have that $x_{\max} = 0.2917$.

This example can be similarly extended for any other value of Q, such that $V \geq Q$. Thus, we prove the *tightness* of our analysis of CCH and BUH, for all values of Q.

APPENDIX C
EXECUTION CYCLES OF BENCHMARKS

The worst-case execution cycles of both benchmarks used in Section 7, are obtained through experimental measurements on the SCC. Each benchmark is executed 1,000 times at every available frequency on SCC, and the highest measured value for each benchmark is used as the worst-case execution time for each frequency.

This section presents Figure 10, which summarizes such experimental measurements of the execution cycles of each benchmark. The figure shows the minimum, average, and maximum (worst-case) measured execution cycles for each frequency.

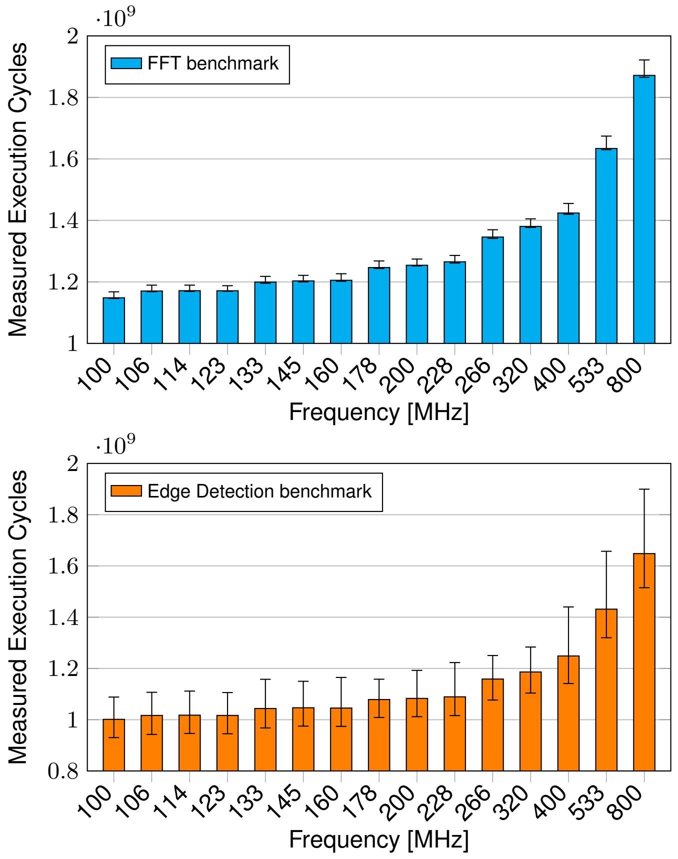


Fig. 10: Experimental execution cycles for benchmarks, where error bars represent the minimum and maximum (worst-case) measured execution cycles, and the bar represents the average execution cycles.

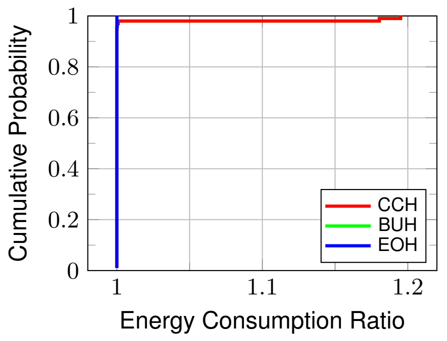
APPENDIX D
DETAILED SIMULATIONS RESULTS

The results of the simulations presented in Figure 9 only show the minimum, average, and maximum energy consumption values among 100 different random cases of synthetic tasks. In this section, Figure 11 presents the same results, but using an empirical cumulative distribution representation. To properly present the results, one sub-figure is needed for every V and Q configuration.

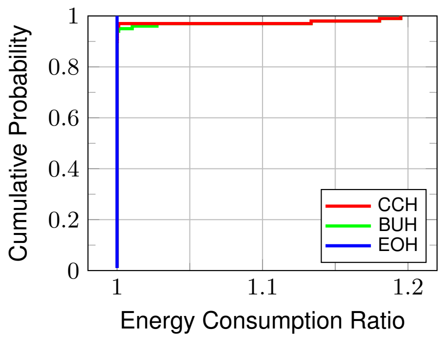
Figure 11 shows that all heuristics derive the optimal solution for more than 80% of the cases of 100 randomly generated workloads. This implies that for general cases, the heuristics have a good performance in terms of energy consumption. Particularly, the smaller the system, the better the heuristics behave. This is an expected result given that, for example in EOH, when the system has just a few cores, the chances for randomly deriving the optimal mapping increase, as seen in Figure 11a where $M = 4$. Nevertheless, there are corner cases for which DYVIA derives better mapping decisions, and the percentage of such cases increases with the value of M.

Symbol	Description
N	Total amount of tasks
$\{\tau_1, \tau_2, \ldots, \tau_N\}$	Set of N periodic real-time tasks with implicit deadlines
p_j	Period (and relative deadline) of τ_j
c_j	Worst-case execution cycles of τ_j
D	Hyper-period: least common multiple among the periods of all the tasks
V	Number of voltage islands in the system
Q	Number of cores per voltage islands
M	Number of homogeneous cores in the system
M'	Number of cores used for task partitioning
$\{\mathbf{T}_1, \ldots, \mathbf{T}_{M'}\}$	Resulting task sets after partitioning
$\{\mathbf{T}_1, \ldots, \mathbf{T}_M\}$	Resulting task sets after partitioning and adding $M - M'$ dummy task sets
w_i	Cycle utilization of task set $\mathbf{T}_i$. Task sets are ordered such that $w_1 \leq w_2 \leq \cdots \leq w_M$
$\{I_1, I_2, \ldots, I_V\}$	Set of voltage islands, ordered with respect to their voltage in an increasing manner
η	Power consumed by each *active* voltage island
$P(s)$	General power consumption function of a core executing a task at frequency s
β'	Power consumed by a core in a low-power mode (not executing anything)
$[s_{\min}, s_{\max}]$	Minimum and maximum frequencies of cores
s_{crit}	Critical Frequency: Minimizes the energy when the overhead for sleeping is negligible
s_j	Frequency of execution (in *cycles per second*) for all cores in voltage island I_j, under SFA
E_j	Energy consumption of island I_j during a hyper-period
LTF	Largest-Task-First strategy, for task partitioning
SFA	Single Frequency Approximation scheme
CCH	Consecutive Cores Heuristic
BUH	Balanced Utilization Heuristic
EOH	Extremal Optimization Heuristic [13]
DYVIA	Dynamic Voltage Island Assignment algorithm
e	Euler's number.
$\mathbf{Y} = \{y_1, y_2, \ldots, y_V\}$	Set of the indexes of task sets with the highest utilizations in each island
$\mathbf{L}_j = \{\ell_{j,1}, \ldots, \ell_{j,Q}\}$	Set with the indexes of the task sets assigned to island I_j
$\mathbf{L} = \{\ell_0, \ell_1, \ldots, \ell_Q\}$	Set of the indexes of the task sets assigned onto cores in one *general* island (as opposed to set $\mathbf{L}_j$)
$H(\mathbf{L})$	Function that computes the energy consumption of an island assigned with task sets $\mathbf{T}_{\ell_1}, \ldots, \mathbf{T}_{\ell_Q}$
$\Lambda(i, j)$	Set of all possible $\mathbf{L}$ sets that satisfy Theorem 2, i.e., the *potentially optimal combinations*
$G(i, j)$	Function that returns the minimum energy for the assignment of task sets $\mathbf{T}_i, \ldots, \mathbf{T}_j$
$BTG(i, j)$	Table with task sets indexes $\{\ell_1, \ldots, \ell_Q\}$ that derive the minimum energy in sub-problem $G(i, j)$
Z	Number of iterations for building table $G(1, M)$
$E_j{}^{\text{RUN=SFA}}_{\text{ASG=ANY}}$	Energy consumption of island I_j that uses *any* mapping algorithm and SFA in individual islands
$E_j{}^{\text{RUN=SFA}}_{\text{ASG=SFA}}$	Energy consumption of task sets $\mathbf{T}_{\ell_{j,i}}$ for $i = 1, \ldots, Q$ in the optimal assignment under SFA
$E_j{}^{\text{RUN=p.c.DVFS}}_{\text{ASG=ANY}}$	Energy consumption of the task sets $\mathbf{T}_{\ell_{j,i}}$ for $i = 1, 2, \ldots, Q$ by using per core DVFS
$\text{AF}^{\text{RUN=SFA}}_{\text{ASG=ANY}}$	Approximation factor for *any* mapping heuristic that activates $\lceil M'/Q \rceil$ islands under SFA
AF_{SFA}	Approximation factor of SFA for the task sets assigned on *a single voltage island*
$E^{\text{RUN=SFA}}_{\text{ASG=DYVIA}}$	Total energy when mapping with DYVIA and using SFA on individual islands
$E^{\text{RUN=DVFS}}_{\text{ASG=DYVIA}}$	Total energy when using DYVIA and an optimal DVFS schedule on individual islands
$E^{\text{RUN=SFA}}_{\text{ASG=DVFS}}$	Total energy when using SFA on individual islands and mapping task sets using an algorithm which is optimal when using an optimal DVFS schedule for executing on individual islands
$E^{\text{RUN=DVFS}}_{\text{ASG=DVFS}}$	Total energy for the optimal mapping and DVFS schedule solution

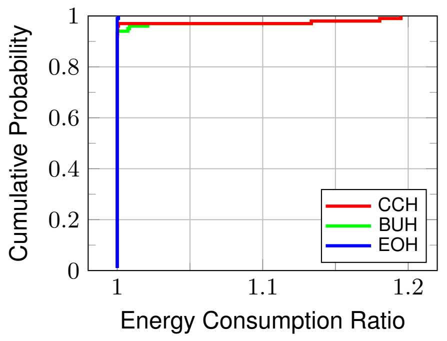
TABLE 6: Table listing all the symbols in the paper.



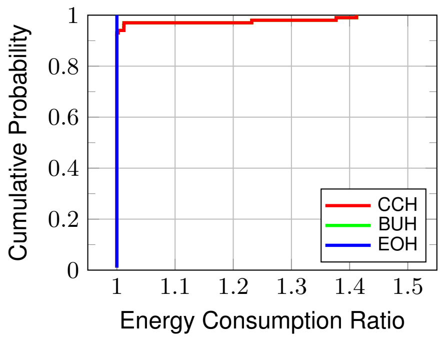
(a) $V = 2$ and $Q = 2$



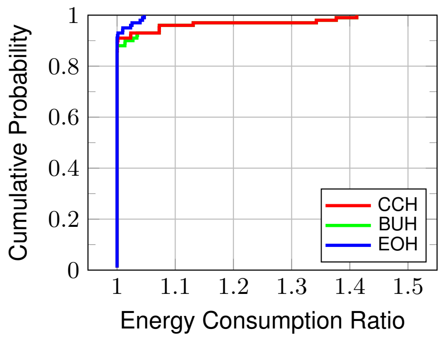
(b) $V = 4$ and $Q = 2$



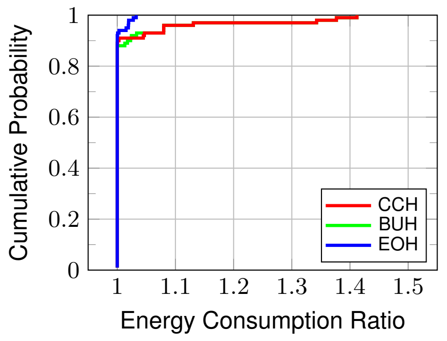
(c) $V = 6$ and $Q = 2$



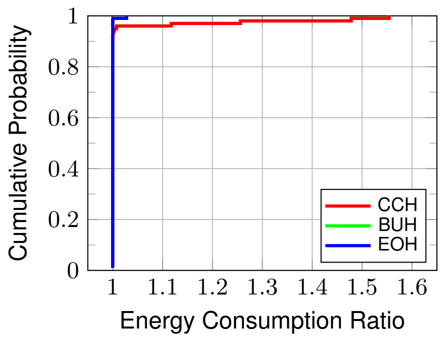
(d) $V = 2$ and $Q = 4$



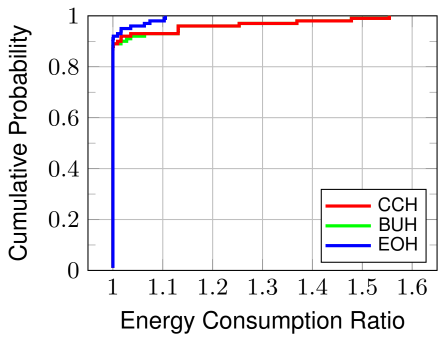
(e) $V = 4$ and $Q = 4$



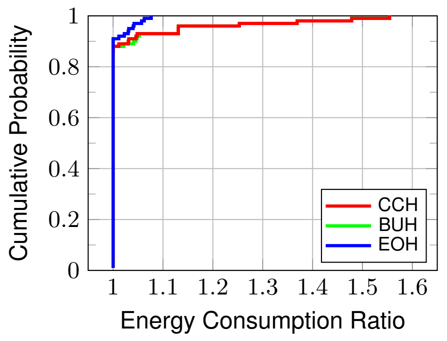
(f) $V = 6$ and $Q = 4$



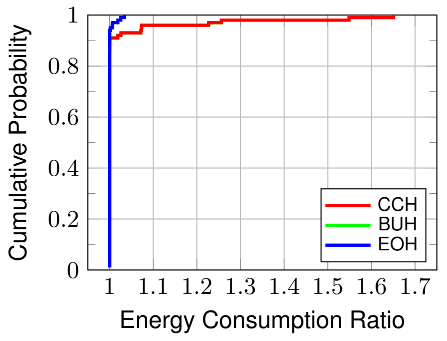
(g) $V = 2$ and $Q = 6$



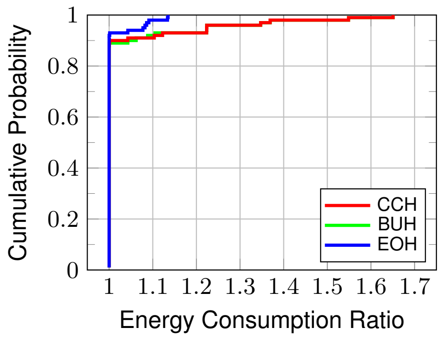
(h) $V = 4$ and $Q = 6$



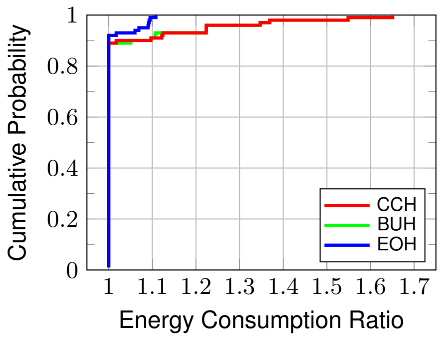
(i) $V = 6$ and $Q = 6$



(j) $V = 2$ and $Q = 8$



(k) $V = 4$ and $Q = 8$



(l) $V = 6$ and $Q = 8$

Fig. 11: Empirical cumulative distribution representation of the simulation results from Section 8.2.

APPENDIX E
MEASUREMENT ERRORS

For our experiments in Section 7, the energy consumption values presented Table 2 are obtained through the integration of power measurements. Specifically, the power consumption of all cores is measured (sampled) every 1 millisecond. This power measurement is then multiplied by the elapsed time from the previous measurement, and added to the total measured energy consumption. We also compute the *expected* energy consumption, and this computation is based on the power profile of our benchmarks presented in Table 1, and considering the experimental lower-bound for the worst-case execution cycles of each task. The values presented in Table 2 show some difference with respect to the expected energy consumptions. These differences are due to several factors, which are already mentioned in Section 7.2. These are: the resolution of the voltage and current meter inside SCC, the presence of one *faulted* core in our SCC platform, the actual execution cycles of each task instance (which could be different from the expected worst-case execution cycles), and the intrinsic integration error from the 1 millisecond resolution in which we measure the total power when computing the consumed energy.

Particularly, the resolution of the voltage and current meter inside SCC result in a 0.3 Watts resolution for each power measurement. This resolution in the power measurements results in some error when conducting the power profile in Table 1, and it also adds error to the energy measurements in Table 2. The *faulted* core results in additional error for both these tables, given that it adds noise to the power measurements, and such noise cannot be easily filtered out. Furthermore, a difference between the actual execution cycles of the tasks and our experimental lower-bound for the worst-case execution cycles (for which the expected energy consumptions are computed) has a clear impact in the energy consumption. This happens because energy is the integration of power through time, and changes in the execution time result in changes in the final energy consumption.

Finally, there is also the integration error. Integration error is intrinsic to any digital energy consumption measurement in which the energy is indirectly computed by measuring power and time. That is, energy is the integration of power through time, where power and time are continuous magnitudes. However, both power are time are here discrete values, with a given resolution for each case. Furthermore, although in a computing system we would expect that the changes in power are synchronized with the execution frequency, thus removing any integration error, this is not the case for our experimental setup. In order to have no integration error, the power measurements should be taken at a frequency no smaller than the highest frequency in which the power changes actually occur. However, in our experimental setup we are unable to measure time with a resolution higher than 1 millisecond. Figure 12 presents an example for measuring energy through digital integration. The power consumption is a sine wave of period 20 millisecond and amplitude of 10 Watts, for which the changes in power happen every 0.2 milliseconds. The sample period in which the power is measured is 1 millisecond.

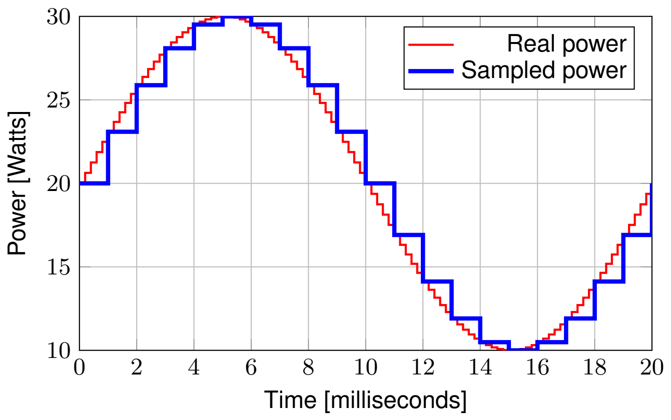


Fig. 12: Integration Error Example. The power consumption is a sine wave of period 20 ms and amplitude of 10 Watts, with changes in power every 0.2 ms. The sample period in which the power is measured is 1 millisecond.