A Fast Heuristic for Tasks Assignment in Manycore Systems with Voltage-Frequency Islands

Shervin Hajiamini*, Behrooz Shirazi*, Hongbo Dong+
*School of Electrical Engineering and Computer Science, +Department of Mathematics
Washington State University
Pullman, WA, U.S.A.
Email: {shervin.hajiamini, shirazi, hongbo.dong}@wsu.edu

Abstract—**Dynamic Voltage/Frequency Scaling (DVFS) is a well-known technique that dynamically scales cores' Voltage/Frequency (V/F) levels to save energy or minimize the application's execution time in manycore systems. This paper proposes an optimization framework that provides a DVFS-based cost- and energy-efficient methodology to balance time-energy tradeoffs in manycore systems with Voltage/Frequency Island (VFI) architectures. The proposed methodology has two steps: 1) formulating a Mixed Integer Linear Programming (MILP) problem that populates islands through the task-to-island assignments given that the islands are symmetric, 2) formulating an Integer Linear Programming (ILP) problem that computes the V/F levels of cores in each island per execution phase of parallel applications. The first step, which is performed at compile-time, considers the per execution phase computational characteristics of the tasks for islanding while the second step optimizes the V/F levels of formed islands at runtime. As solutions time of the proposed task-to-island assignment problem increases significantly with a large number of tasks or islands, this paper presents a fast heuristic that only requires a sorting procedure in its most time-consuming step and obtains near-optimal solutions. The proposed framework's energy efficiency is compared to an optimal, per-core islanding that establishes the best energy-time solutions for the experimented applications. Using Energy-Delay Product as performance metric, experimental results show that the framework's energy efficiency, at the worst case, is within 13% of the per-core DVFS. The results also show that this framework utilizes the idle times of low Central Processing Unit (CPU)-intensive benchmarks to increase energy saving with reasonable performance loss.**

Keywords-Manycore System; Task Partitioning; Dynamic Voltage-Frequency Scaling; Voltage-Frequency Islands; Optimization Framework; Energy Efficiency.

I. INTRODUCTION

Large-scale computer systems have become more pervasive by providing computing resources to solve complex applications. Parallel computing utilizes the multiprocessing aspect of the computing resources (e.g., CPU cores) to perform simultaneous computational processes (tasks) in order to increase the speed of the system. To strengthen the Operating System (OS) capability for running the user tasks in parallel, applications are instrumented by parallel programming techniques to take advantage of the increasing cores' computing power, which are interconnected and used as shared resources within a single computer system. As the number of cores continues to scale in manycore systems, excessive energy consumption has become a primary concern for the system designers and devising effective energy-aware techniques that are sustainable with the applications' computational demands is an important research area.

The Dynamic Voltage and Frequency Scaling (DVFS) is a method for executing high performance applications on manycore systems while maintaining the system energy consumption below a user-defined energy budget [6]. There are three approaches to apply the DVFS for energy efficiency optimization in the manycore systems. (1) Running an application on a chip-wide DVFS, where a common Voltage/Frequency (V/F) level is assigned to all the cores [1]. This method does not scale with the varying applications' computational demands. (2) In the per-core DVFS approach, the V/F level for each core is adjusted throughout the program execution, resulting in the best energy efficiency, but at the cost of hardware complexity and complicated system level control [2]. (3) As a compromise, a more flexible Voltage and Frequency Island (VFI) approach has been adopted, where cores in an island share the same V/F level, which may vary during the program execution based on the program characteristics [3].

Nowadays, large high performance computing applications have changing computational behavior during the applications runtime. Fixing the VFIs' V/F levels for the entire application execution limits exploiting opportunities to speed up or down the islands speed to gain high performance or energy saving depending on the application characteristics [8]. To address this limitation, this work applies the DVFS on islands where each VFI's V/F level can be configured individually during the program runtime.

The traditional approach for islanding is to group cores executing similar tasks across the application's execution phases (intervals) [14]. A more effective approach to perform the partitioning is to identify the similarities of tasks within the individual execution phases of the applications. This way, the system energy efficiency can be further improved by incorporating the tasks computational variations, across and within the execution phases, into the problem's optimization objectives.

The VFIs may have the same size or number of cores (symmetric) or may be asymmetric in size [4]. To simplify the task-to-island assignment problem, this paper assumes a symmetric system, where the VFIs sizes (the number of cores in a VFI that execute the assigned tasks) are the same.

This paper presents a framework for optimizing the task-to-island assignments (tasks partitioning) and the VFIs' V/F level assignments. Using this framework, this paper's goal is to minimize the applications' total execution times (makespans) without exceeding user-defined energy budgets/limits.

The framework proposed in this paper has the following contributions compared to our previous work [4], which also discussed a method for an energy-constrained, optimized makespan VFI-based system:

- The VFIs' V/F levels are dynamically changed per execution phase of the experimented applications.
- The same V/F level can be assigned to different VFIs in each execution phase to achieve an overall better energy/time tradeoff.
- To improve the system's energy efficiency, application tasks, with similar computational characteristics, are assigned to the VFIs before applying DVFS.
- This paper demonstrates the extent to which the energy efficiency is maximized considering the applications with different compute/memory intensive workloads.
- Compared to [4], the proposed heuristic is faster and more scalable for larger applications or system sizes.

This paper is structured as follows. Section II summarizes related works followed by our contributions. Section III describes a system model and a model for executing applications on the system. Section IV explains the proposed two-step framework. Section V and Section VI present experimental setup and the results, respectively. Section VII concludes this paper.

II. RELATED WORK

The multi/manycore processing is a form of parallel computing where a parallelized application uses the shared hardware resources (CPU cores) to simultaneously execute the applications' threads and shorten the application runtime [5]. Increasing the number of cores in a chip may improve the application speedup but it overheats the chip due to the energy consumed by cores during the idle and busy periods. The DVFS is a well-known method that has been used to address this problem with two mainstream techniques. The per-core DVFS is a resource-demanding technique where a separate voltage regulator is allocated to each core to adjust its V/F level at runtime (lowering energy during idle periods and increasing it during compute phases). As a second method, the VFI-based systems provide a less complex and economical alternative where the V/F level of an island of cores is tuned by a single regulator. The VFI-based systems are cost-effective and provide reasonable energy saving opportunities with acceptable application execution delay [6][7]. The following summarizes the VFI-based works that are related to this paper.

The VFIs' V/F levels are determined either statically (at compile-time) or adjusted dynamically (at runtime) to account for the applications' computational variations. For example, Duraisamy et al. [8] used the cores' number of instructions per cycle and inter-core data transfers for per VFI static V/F level assignment, while Ogras et al. [9] used a feedback controller to dynamically adjust the V/F levels of a Network-on-Chip (NoC)-based VFI system using the occupancy levels of inter-VFI queues.

In terms of VFIs formation, both the symmetric and asymmetric partitioning of cores has been deployed. David et al. [10] partitioned 24-tile Intel's single-chip cloud computer into 6 VFIs, each one containing 4 tiles (symmetric). Jin et al. [11] used asymmetric VFIs whose sizes are reconfigured once by adding cores that were not assigned to the same VFI through multiple static optimizations of VFIs formation.

The prior research works have solved one or both of the islanding and V/F level assignment problems. Ozen et al. [12] used two VFIs with corresponding fixed V/F levels in a NoC, where cores' slack times were used to run the under-loaded VFIs with lower V/F levels to minimize the energy consumption. Ogras et al. [13] performed the islanding and V/F level assignment iteratively by merging two VFIs, which resulted in reducing the system energy consumption while maintaining the performance constraints.

The islanding and V/F level assignment problems have been solved by heuristics or linear programming-based (LP) techniques. Ghosh et al. [14] used ILOG CPLEX, an Integer LP-based technique, for determining the physical locations of cores on NoC-based VFIs and their respective V/F levels. Jin et al. [15] used a statistical heuristic that used the probability distributions of the tasks' execution times and energy consumptions under different V/F levels. The VFIs' V/F levels were determined such that tasks with large energy variations are assigned more slack and run with lower V/F level to maximize the energy saving.

A number of works have addressed the task scheduling (or task assignment) when formulating energy efficiency objectives for systems with homogenous and heterogeneous compute nodes. Leung et al. [17] proposed a list scheduling algorithm to compute the tasks priorities, executed on NoC-based equally-sized islands, based on the links communication delays. Chou et al. [18] devised an iterative task mapping heuristic that identified and grouped the neighboring idle cores of a NoC, with pre-defined V/F levels, for the application tasks assignment. Oxley et al. [19] analyzed the robustness of a set of heuristics, used for the static assignment of tasks to heterogeneous nodes, in terms of meeting makespan deadlines or energy budgets considering the stochastic tasks execution time.

The research contributions of this paper include:

- Formulating a MILP for the task-to-island assignment problem that forms the symmetric islands of tasks with similar computation behavior. In a sense, the proposed formulation aims at forming per execution phase islands based on measuring the tasks characteristics for each execution phase of the applications.
- Formulating an ILP for the VFIs' V/F level assignment problem that performs DVFS on the islands per execution phase in order to minimize the applications makespans under the user-defined energy budgets.
- Proposing a fast and low-cost heuristic to solve the task-to-island assignment (tasks partitioning) problem. The experimental results show that when

using the heuristic for task-to-island assignments, the system energy efficiency, measured by the Energy-Delay Product (EDP) metric, is, at worst, within 13% of the optimal per-core DVFS across the experimented benchmarks. Furthermore, the results show that the proposed framework efficiently maximizes the energy saving of low CPU-intensive benchmarks.

III. MANYCORE SYSTEM CONFIGURATION

This section presents assumptions about the multiple-VFI manycore system setup and the execution model of applications running on this system. This section also explains an applications profiling strategy that provides the task-level application characteristics that are utilized by the VFI-based optimization framework to measure the energy-performance tradeoff.

A. *VFI-based Manycore System Design*

This paper assumes an N-core manycore system $C = \{c_1 \dots c_N\}$, where cores are arranged in a $\sqrt{N} \times \sqrt{N}$ mesh of homogenous cores. It is assumed that the system is partitioned into a fixed number of symmetric islands, $I = \{i_1 \cdots i_K\}$ where there are $Q = N/K$ cores per island. For example, Figure 1 shows a partitioned system with $K = 3$. Also, $Q = 1$ represents a manycore system with the most fine-grained islands. The cores in a VFI operate under a common V/F level, which is determined by the V/F level assignment step of the framework. These V/F levels are attained from a range of available CPU performance states: $S = \{s_1 \dots s_L\}$ where s_1 and s_L correspond to the lowest and highest V/F levels, respectively. Any two VFIs may have the same or different V/F levels, which impact the system's overall energy efficiency. Each core has a local non-unified L1 cache and all cores share a unified L2 cache.

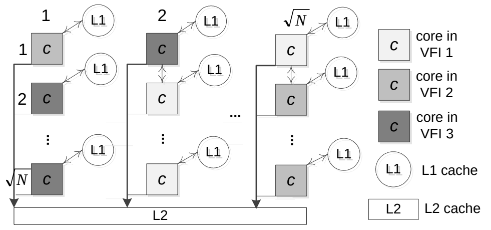

Figure 1. A manycore system with three islands

B. *Application Execution Model*

This paper considers multithreaded applications chosen from benchmark suites, which will be explained in Section V, where each thread of execution runs on a particular core and is not re-assigned to another core during the application execution. The execution of these applications follows Single Program Multiple Data (SPMD) parallelization technique wherein the same program is split up among cores to perform tasks on different data. These benchmarks are developed and utilized in a shared memory system that facilitates inter-core/thread data exchange at runtime [16]. The execution runs, which are used to evaluate the optimization goals, encompass a unique section inside the benchmark's source codes known as Region Of Interest (ROI).

ROIs, representing the parallel sections of the applications, are divided into multiple tasks, according to the SPMD model, and are assigned to cores/threads for the parallel execution. Because of the changing workloads of the applications (benchmarks), the execution of the ROIs represents distinct application characteristics in the form of phases or execution windows during the runtime. During the applications execution, some of threads produce data while the others consume it. To ensure that the consumer threads obtain the correct data before executing the next phase of the applications, the benchmarks' ROIs are instrumented by synchronization routines (such as barriers), which resolve, among the cores, data memory access delays within the phases, as well as data transfers across the phases of the applications. The execution of a number of instructions between two consecutive synchronization points defines a distinct computational phase of the benchmark, which are represented as the cores' parallel tasks within that execution phase. Figure 2 shows an example of an application with P execution phases where within each phase gray portions show the computation periods of cores executing their tasks and black portions show the core's idle periods. These periods, representing execution overheads, may be created by memory access delays (or data transfers) resulting in idle periods upon reaching synchronization points at the end of each phase.

<u>Task model</u>

An application consists of a set of tasks sets $T = \{T_1 \dots T_P\}$ defined over the P execution phases where T_j denotes a task set executed in phase j $(1 \leq j \leq P)$ of the application. Each task set T_j is composed of tasks executed by cores in the corresponding application phase where $\tau_{j,i}$ denotes task i $(1 \leq i \leq N)$ in phase j. Thus, it is assumed that each core executes one task in the application phase. As indicated above, the execution of a task set in the next phase is dependent on the completion of a task set in the previous phase. As such, the assignment of tasks to islands represents typical application task graphs, assuming a negligible/zero memory access delays between the dependent tasks (because the memory access delays for data transfers among the task sets are already accounted for in the tasks execution time).

The tasks partitioning formulation considers the similarity of the tasks' workloads in an execution phase to perform the task-to-island placements. The outcome of the task-to-island assignments guides the VFIs' V/F level assignment formulation to improve the system's energy efficiency by slowing down VFIs with lower workloads and speeding up the highly loaded VFIs.

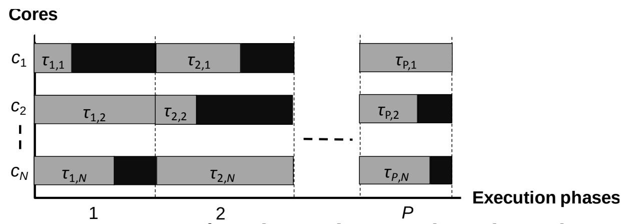

Figure 2. Execution of a P-phase application with N tasks per phase

C. *Application Profiling Methodology*

The optimization framework has a priori knowledge of the benchmarks/applications execution. The profiling data used for the static optimization of the task-to-island assignments and their V/F levels include the execution times, energy consumptions, and workloads of the task set for each execution phase of the benchmark collected at each possible V/F level. This paper uses a profiling strategy that runs the benchmark on the manycore system once per V/F level and collect the pertinent per phase execution time, energy usage, and workload information of all tasks in that phase. Here, the execution time corresponds to the computational period of a task in the execution phase before reaching the barrier (black portions in Figure 2). The energy consumption means the rate of the task power usage during its execution in the corresponding phase. The workload is defined as the ratio of the task's busy (computation) cycles to the total cycles (the summation of the busy and idle cycles) in the execution phase.

IV. TWO STEP TASK-TO-ISLAND ASSIGNMENT AND V/F LEVEL ASSIGNMENT TECHNIQUE

The task-to-island assignment and V/F level assignment steps are formulated in this section. To reduce the computation time of solutions obtained by the optimization framework, this paper solves the above steps sequentially. The islanding step uses the tasks' workloads to identify the groups (islands) of tasks with similar computational similarities per execution phase. The V/F level assignment step considers the execution time and energy usage of the islands under multiple V/F levels to make the best performance-energy tradeoff that minimizes the benchmarks makespan given an energy budget.

A. *Task-to-Island Assignment*

As mentioned above, partitioning the tasks among the islands is based on the similarity of tasks. To measure the degree of similarity among tasks, this formulation computes the percentage difference ratio between a task workload and the maximum workload in an island to which the task may be assigned. To find the maximum similarity among the tasks, this optimization step minimizes the ratio that indicates the wasted workload.

The following are the problem's objective and constraints:

$$\underset{y_{i,k},F_k,c_k,z_{k,j},x_{i,k}}{Minimize} \quad Y_j = \sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k} \quad \forall T_j \in T \tag{1}$$

$$y_{i,k} \geq x_{i,k} - t_i \cdot F_k \quad \forall \tau_{j,i} \in T_j, \forall i_k \in I \tag{2}$$

$$c_k = \sum_{j=1}^{r} a_j \cdot z_{k,j} \quad \forall i_k \in I \tag{3}$$

$$F_k = \sum_{j=1}^{r} \left(\frac{1}{a_j}\right) \cdot z_{k,j} \quad \forall i_k \in I \tag{4}$$

$$t_i \cdot x_{i,k} \leq c_k \quad \forall \tau_{j,i} \in T_j, \forall i_k \in I \tag{5}$$

$$\min(t_1 \cdots t_N) \leq c_k \leq \max(t_1 \cdots t_N) \quad \forall i_k \in I \tag{6}$$

$$\sum_{k=1}^{K} x_{i,k} = 1 \quad \forall \tau_{j,i} \in T_j \tag{7}$$

$$\sum_{i=1}^{N} x_{i,k} = Q \quad \forall i_k \in I \tag{8}$$

$$\sum_{j=1}^{r} z_{k,j} = 1 \quad \forall i_k \in I \tag{9}$$

$$z_k = \{z_{k,1} \cdots z_{k,r}\} \quad \forall i_k \in I \text{ (SOS-2 variable set)} \tag{10}$$

$$y_{i,k} \geq 0 \quad \forall \tau_{j,i} \in T_j, \forall i_k \in I \tag{11}$$

$$F_k \geq 0 \quad \forall i_k \in I \tag{12}$$

$$c_k \geq 0 \quad \forall i_k \in I \tag{13}$$

$$x_{i,k} \in \{0,1\} \quad \forall \tau_{j,i} \in T_j, \forall i_k \in I \tag{14}$$

$$z_{k,j} \geq 0 \quad \forall i_k \in I, 1 \leq j \leq r \tag{15}$$

The task-to-island assignment formulation aims at minimizing the wasted workloads of islands for every execution phase. The island's maximum workload is not known before solving the above optimization problem. Therefore, the problem objective (the percentage wasted workload) becomes non-linear. The non-linear functions are typically linearized to obtain optimum solutions more efficiently. The non-linear curve of a function representing the island's maximum workload, is linearized by a mathematical technique, known as the piece-wise linear function [19], which approximates the actual value of the non-linear function. For the linearization, this technique divides the function's non-linear curve (such as the objective function in this paper) into multiple segments of straight lines that each can be represented by a linear function.

Y_j denotes the total wasted workload in execution phase j. $y_{i,k}$ is the wasted workload of task $\tau_{j,i} \in T_j$ ($1 \leq i \leq N$) in island i_k. c_k is the approximation of island's maximum workload. F_k approximates $1/c_k$. These approximations use Special Ordered Set (type 2) variables (SOS-2), $z_{k,j}$, where each variable indicates how likely it is that a line segment, connected by two adjacent points (i.e., a_j and a_{j+1}), approximates c_k or $1/c_k$. Technically, the SOS-2 variables transform the piece-wise linear functions to a form that can be used by linear programming methods to solve optimization problems. t_i is the workload of task $\tau_{j,i}$. $x_{i,k}$ shows where task $\tau_{j,i}$ is assigned to island i_k. Q denotes the number of tasks assigned per island. r is the number of adjacent points that form the line segments.

Constraint (1) minimizes the total amount of wasted workloads for a task set across all the islands. Constraint (2) computes the wasted workload if a task is assigned to an island. Constraints (3) and (4) approximate c_k and $1/c_k$, respectively. Constraint (5) determines c_k (the maximum workload of an island). Constraint (6) ensures that the island's maximum workload is within the minimum and maximum values of tasks workload in an execution phase. Constraint (7) shows that a task is assigned to only one island. Constraint (8) indicates that all islands have an equal size. For all the SOS-2 variables defined in (10), only two of them are non-zero. These non-zero variables, which have to be adjacent, indicate the two end points of a line segment.

Figure 3 shows an application running on a system with 2 execution phases (P = 2) and 4 tasks per phase ($|T_j|$ = 4, $1 \leq j \leq 2$) before (3(a)) and after (3(b)) applying the task-to-island assignment formulation. For two symmetric islands (K = 2), it is observed from 3(b) that in the first execution phase, $i_1 = \{\tau_{1,1}, \tau_{1,3}\}$ and $i_2 = \{\tau_{1,2}, \tau_{1,4}\}$ whereas for the second execution phase, $i_1 = \{\tau_{2,1}, \tau_{2,4}\}$ and $i_2 = \{\tau_{2,2}, \tau_{2,3}\}$. For example, for the first phase in Figure 3, the wasted workload, Y_1, is computed based on i_1 and i_2 where the task pair in each island has the most similar execution workloads. It should be noted in Figure 3 that i_1 and i_2 can be executed on any combination of 4 cores in each execution phase because 1) it is assumed that the system consists of homogenous cores, and 2) the islanding is performed independently per execution phase due to the synchronization of threads at the end of the phase.

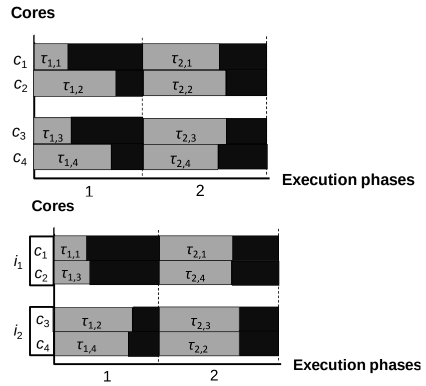

Figure 3. Application task sets with N = 4 and K = 2 showing (a) default and (b) optimized tasks assingment

B. VFIs' V/F Level Assignment

The goal of islanding step, discussed above, is to separate the islands with different workloads using the tasks computational similarity. For a given V/F level, any two islands with different workloads may have different execution performance. Such performance gap among the islands is utilized by the V/F level assignment step to maximize the system energy saving while increasing the performance within the allocated energy budget. This is accomplished by slowing down islands with low workloads and speeding up the ones with high workloads.

Running the islands (VFIs) under the fixed V/F level for the entire application execution may improve energy-performance tradeoff for applications with steady workloads but it has poor performance outcomes for applications with changing workloads at runtime. The second step in the optimization framework addresses this concern by adjusting the islands' V/F levels per execution phase of applications based on the workloads intensity of islands in the corresponding application phase.

The following are the objective and constraints for formulating the V/F level assignment problem:

$$\underset{\theta_j, a_{k,l,j}}{Minimize} \quad \theta = \sum_{j=1}^{P} \theta_j \tag{16}$$

$$\sum_{l=1}^{L} d_{k,l,j} \cdot a_{k,l,j} \leq \theta_j \quad \forall i_k \in I, \forall T_j \in T \tag{17}$$

$$\sum_{l=1}^{L} a_{k,l,j} = 1 \quad \forall i_k \in I, \forall T_j \in T \tag{18}$$

$$\sum_{j=1}^{P} \sum_{k=1}^{K} \sum_{l=1}^{L} e_{k,l,j} \cdot a_{k,l,j} \leq EB \quad EB \geq 0 \tag{19}$$

$$a_{k,l,j} \in \{0, 1\} \;\; \forall i_k \in I, \forall T_j \in T, \forall s_l \in S \tag{20}$$

Where, Θ is the makespan of application. Θ_j is the execution time of phase j, which is determined by the maximum finish time among islands in that phase. $d_{k,l,j}$ and $e_{k,l,j}$ are the execution time and energy consumption of a core running a task, assigned to VFI i_k, under V/F level l at the execution phase j, respectively. $a_{k,l,j}$ states whether the V/F level l is assigned to i_k in phase j. EB constrains the system energy consumption for the application execution.

The problem objective (16) minimizes the benchmark's makespan, defined by the execution times of application phases. Constraint (17) determines the execution time of a phase. Constraint (18) affirms that only one V/F level is assigned to an island per execution phase. Constraint (19) ensures that the system's energy consumption, computed by the energy usage of VFIs across all execution phases, does not exceed the user-defined energy budget.

Figure 4 depicts an example of V/F level assignment step for the same application task sets shown in Figure 3. It is observed from Figure 4 that in the first execution phase, V/F levels s_2 and s_4 are assigned to i_1 and i_2, respectively. Since i_1 has a lower computational workload than i_2 in the first phase, running it with the lower V/F level (s_2) results in saving more energy while running i_2 with the higher V/F level (s_4) improves the performance. For the second execution phase, i_1 and i_2 have comparable workloads. Thus, s_3 is assigned to both islands.

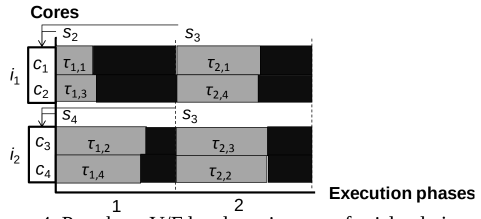

Figure 4. Per phase V/F levels assignment for islands i_1 and i_2

C. Fast Heuristic for Task-to-Island Assignment

The task-to-island assignment problem is an NP-hard problem due to its growing complexity when experimenting with larger task sets size or the number of islands per execution phase. To reduce the computation time of solving this problem, this section presents a fast, practical heuristic that only requires a sorting procedure in its most time-consuming step.

This heuristic performs the following two steps per application phase: 1) tasks are sorted in the increasing order of their execution workloads. In other words, the sorting procedure orders the tasks (i.e., from small to large tasks) based on their computational workloads. As mentioned in

Section III-C, the execution workload refers to the task utilization measured over an application phase's time span and is computed as the ratio of the core's busy cycles to the total execution cycles. The utilization values do not significantly change when running cores/islands with different V/F levels at runtime. Therefore, this performance measure was chosen for the task-to-island assignments in each execution phase. 2) every $Q = N/K$ consecutive sorted tasks are assigned to an island (N and K are number tasks and islands, respectively). The time complexity of step (1) increases with $O(N \cdot \log(N))$ in the best case while step (2) is performed in constant time.

It should be mentioned that the assignment of tasks to islands implies that Q cores are allocated to the corresponding Q tasks assigned to an island because the application execution model (see Section III-B) assumes that a core executes only one task in each phase.

D. Real-life Realization of VFI-based System

The application of the proposed optimization framework in embedded systems is useful when multicore processors are designed to run specific applications many times given system configurations that are pre-optimized once at compile-time. To use this framework for such cases, the applications are first profiled using the profiling method explained in Section III-C. At compile-time, the islanding step assigns tasks to islands and the V/F level assignment step determines the VFIs' V/F levels. The per VFI, per execution phase V/F levels are then stored in a look-up table to be used later at runtime when at the start of each execution phase the OS fetches the V/F levels from the table and uses special registers to communicate the V/Fs with DVFS controllers that tune the islands' performance.

V. Experimental Setup

To measure the energy efficiency of the proposed framework, General Execution-driven Multiprocessor simulator (GEM5) [20], a full-system simulator, is used to model 64 cores that are arranged as a 8x8 mesh structure of homogenous cores, where each core has 64KB L1 instruction and data caches and a shared 8MB L2 cache. All the benchmarks are run 4 times using the following V/F levels: s_1: 0.5V/ 1.25GHz, s_2: 0.667V/ 1.666GHz, s_3: 0.834V/ 2.083GHz, s_4: 1.0V/ 2.5GHz, which are within a nominal range of states that provide stable performance and power data. The per execution phase task sets workload and execution time are collected as explained in Section III-C. To obtain the phases' energy consumption, the GEM5's performance outputs are fed to Multicore Power, Area, and Timing (McPAT) [21] that generates the energy consumption for the task sets. The time/energy overheads caused by V/F level switching are not incorporated in the optimization objectives and constraints because they are only about a few hundreds of nano seconds/Joules order of magnitude [6].

The proposed two-step optimization framework is tested on three benchmarks, namely Fast Fourier Transform (FFT), Lower and Upper triangular matrices (LU), and Cache-Aware Annealing (CANNEL) [22][23]. These benchmarks are used in different application domains and represent applications with high or low CPU-intensiveness: the percentage of compute intensity of FFT, LU, and CANNEAL is 96%, 92%, and 85%, respectively where FFT and CANNEAL are high and low CPU-intensive benchmarks, respectively.

Similar to [8], the 64-core system, used in this paper, is partitioned into 4 islands (K = 4) where each island has 16 (Q = 16) tasks, whose assignments to islands are defined by the islanding formulation in Section IV-A. This configuration was chosen to assign sufficient tasks per island in each execution phase.

The formulations, discussed in Section IV, are implemented with a modeling language, Algebraic Language for Mathematical Programming (AMPL) [24], which is used for modeling large-scale constrained optimization problems. To find solutions that make the best energy-performance tradeoff, Gurobi [25], a solver included in the AMPL software package, is used to solve the islanding and V/F level assignment problems. The heuristic is implemented and solved in MATLAB. All experiments for the symmetric VFI-based system are conducted on a CentOS workstation with Intel dual Core x86, 3.3 GHz processor and 3.6 GB RAM. The time and energy usage of workstation's physical cores when running AMPL/Gurobi are not included in the formulations since the problems are solved pre-runtime.

VI. Experimental results

The experimental results consist of four parts. The first part presents the performance (execution time) of benchmarks under the proposed VFI-based optimization framework compared to the optimal performance obtained by the per-core DVFS VFIs. The second part demonstrates the framework's impact on system energy efficiency using two well known metrics. The third part explains the VFIs' V/F level assignment outcomes. The fourth part discusses the optimality of heuristic islanding and VFIs' V/F level assignments.

Figure 5 and Figure 6 refer to the per-core DVFS as Fine-Grained (FG) since $K = N$ (K and N are the number of islands and tasks, respectively) and dynamically tuned VFI system as DCG (Dynamic Coarse-Grained) because the V/F levels of a group of cores are adjusted per execution phase. To constrain the energy budget, the MILP-based formulation considers three levels for EB (19): High (EB(H)), Medium (EB(M)), and Low (EB(L)), which correspond to 7.5%, 22.5%, and 37.5% energy reductions from the benchmarks' energy consumption when all cores run at the fastest V/F level (s_4 in Section V).

There is a large body of research that use (meta) heuristics, greedy, and machine learning techniques for assigning tasks to cores and determining the cores' V/F levels to obtain the best objective values [26]. Instead of comparing the proposed framework performance to such a wide range of existing techniques in the literature, it is compared to the per-core DVFS, which is considered as the most energy-efficient method in high performance computing platforms. Moreover, the degree to which the VFI-based system's energy efficiency is close to the per-core

DVFS indicates how close the proposed framework's outcomes are to the optimal solutions.

A. *Execution Time Comparison*

The ILP-based formulation minimizes the performance (16) of running symmetric coarse-grained islands under the energy budget levels. Figure 5 evaluates DCG vs. FG performance (execution time) relative to the non-DVFS baseline, when all cores operate at the fastest V/F level (s_4), using the following criteria:

1) *Energy Budget*

Intuitively, decreasing the energy budget increases the benchmarks execution times because the islands are slowed down to consume less energy below the energy budgets. Interestingly, for EB(H) in Figure 5, the performance of DCG is comparable to FG. The reason is that for EB(H) the execution time of islands with high workloads dominate the execution time of under-loaded ones. Thus, scaling up the V/F levels of highly loaded islands in DCG improves the system performance while slowing down the under-loaded islands not only has a negligible impact on the overall benchmark execution time but also increases energy saving. By further decreasing the energy budget, the highly loaded islands have to run slower, resulting in a noticeable execution time increase for EB(M) and EB(L).

2) *Benchmarks CPU-intensiveness*

Regarding the impact of benchmarks CPU intensity on DCG, Figure 5 shows that for CANNEAL the system performance penalty stays below 18% across the energy budgets. This is due to the low CPU-intensiveness of CANNEAL whose execution time is not degraded by lowering the energy budget. As such, for CANNEAL, the DCG performance is closer to FG compared to FFT and LU. Since LU has low CPU-intensiveness in some phases, it is observed from Figure 5 that in EB(H) DCG performance is close to FG. Clearly, for a CPU-intensive benchmark like FFT, DCG has the poorest performance when the VFIs run slower in the lower energy budgets.

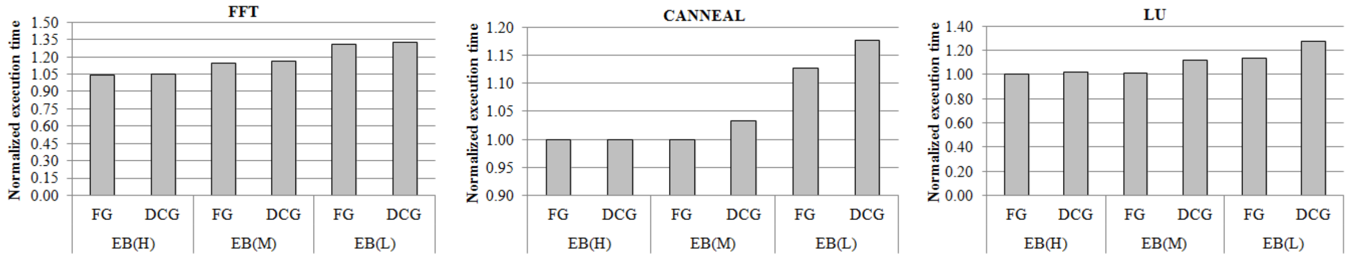

Figure 5. Execution time of Fine-Grained (FG) and Dynamic Coarse-Grained (DCG) system configurations over High (H), Medium (M), and Low (L) Energy Budgets (EB). The execution times are normalized to non-DVFS baseline.

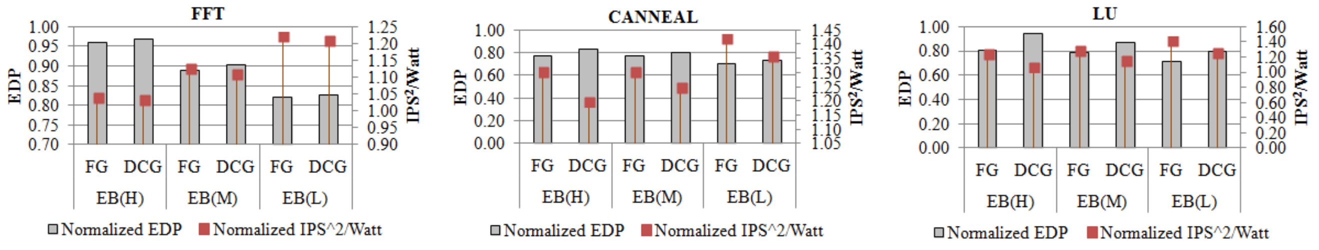

Figure 6. EDP and IPS^2/Watt of Fine-Grained (FG) and Dynamic Coarse-Grained (DCG) system configurations over High (H), Medium (M), and Low (L) Energy Budgets (EB). The EDP and IPS^2/Watt are normalized to non-DVFS baseline.

B. *Energy Efficiency Metrics Comparison*

Besides measuring the framework impact on application performance, the following metrics are used to evaluate the system energy efficiency: 1) Energy-Delay Product (EDP) and 2) Instructions Per Second, per Watt (IPS^2/Watt) [27]. The former measures the amount of energy saving obtained despite performance loss while the latter specifies the amount of throughput gained in exchange for consuming power for running a number of instructions in a time period (e.g., execution phase in the application model). Lower values for EDP and higher values for IPS^2/Watt are desirable.

Figure 6 shows the framework impact on EDP and IPS^2/Watt resulting from the application of the FG and DCG configurations normalized to the corresponding EDP and IPS^2/Watt of non-DVFS for the same benchmarks.

Figure 6 suggests that CANNEAL, compared to FFT and LU, obtains the best (lowest) EDP across the energy budget. Especially, in EB(H), DCG utilizes the CANNEAL's memory access times to maximize energy saving without losing performance while, as a CPU-intensive benchmark, most of the FFT's execution run consists of floating-point instructions, which provide less opportunity for energy saving and cause the EDPs of FG and DCG to be close to one another in EB(H). For LU, compared to FFT and CANNEAL, the EDP gap between FG and DCG is larger, which can be explained by the LU's workload variations across its execution phases. Overall, the average EDP improvements of DCG, compared to non-DVFS, are within

1%, 5%, and 13% of the best EDP improvements obtained by FG for FFT, CANNEAL, and LU, respectively.

IPS^2/Watt is inversely proportional to EDP. Thus, the relative energy efficiency of FG and DCG in terms of IPS^2/Watt is similar to EDP. Figure 6 specifies finer scaling range for the IPS^2/Watt axis compared to EDP to show a clearer difference between the energy-efficient solutions obtained by these two configurations across the energy budgets. Of note, in Figure 6, the upper bound limit of EDP axis is set to 1 to show the EDP improvements against the non-DVFS baseline across the studied benchmarks.

C. VFIs' V/F Levels

As mentioned in Section IV-B, the V/F assignment step tunes the VFIs performance to increase the system energy saving by lowering the V/F levels of less loaded islands and increasing the V/Fs for the heavily loaded ones. The extent to which the islands V/F levels are scaled up or down, depends on the overall characteristics of benchmarks.

Table I shows the V/F states distribution among all islands and across all the execution phases of FFT, LU, and CANNEAL at the high energy budget (EB(H)). This table suggests that the highest V/F level (s_4) constitutes the largest percentage of assignments for FFT and LU (68% for FFT and 61% for LU). This observation matches the high CPU-intensiveness of these benchmarks having highly loaded islands and their V/F states are scaled up to maximize the performance. On the other hand, for CANNEAL, a lower V/F state (s_3) is assigned to 65% of islands, which again corroborates with the low CPU-intensiveness of CANNEAL since lowering V/F levels for such benchmarks saves energy without significant performance loss. Table I also shows that for LU s_1 and s_2 are used for the V/F assignment. That's because some execution phases of LU have less amount of computation, which are utilized by the V/F level optimization step for slowing down the VFIs and saving energy.

TABLE I. PERCENTAGE OF V/F LEVELS ASSIGNED TO VFIS

Benchmark	s_1	s_2	s_3	s_4
FFT	0	0	31	69
LU	16	10	12	62
CANNEAL	0	0	65	35

D. Optimality Analysis of Solutions obtained by Heuristic and ILP-based Formulation

Section IV-C explained a heuristic for the MILP-based formulation of islanding problem. To find out the extent to which the heuristic solutions are close to optimal, the MILP-based formulation, which provides optimal solutions, is solved for a number of execution phases of the experimented benchmarks. To solve the associated problems, the heuristic task-to-island assignments (Section IV-C), are used as initial solutions. For larger problems size, the experiments are run for a week after which it was observed that the differences of solver's objective values (1) were negligible (less than a percent) compared to the objective values obtained by the heuristic and used as the initial seeds to solve the MILP-based formulation. Considering such minimal difference, the islands, obtained by the MILP-based formulation and heuristic, were found to be identical, indicating that the proposed heuristic performs optimally to solve the islanding problem. For N = 64, K = 4, and r = 10 used for Section IV-A, the MILP-based formulation has 560 variables and 644 constraints per application's execution phase.

The computation complexity of solving ILP-based V/F level assignment problem (Section IV-B) depends on the number of islands (K), number of V/F levels (L), and number of execution phases (P). To solve the V/F level assignment problem for DCG (coarse-grained VFIs), K = 4, L = 4, and P is set to 8, 15, and 31 for FFT, LU, and CANNEAL, respectively. Using the above parameter values, the ILP-based problems are optimally solved within a minute, from which the associated performance and energy efficiency results are obtained as shown in Figure 5 and Figure 6.

VII. CONCLUSION

This paper presented a framework that optimizes the tasks partitioning and VFIs' V/F levels to minimize the benchmarks makespan without exceeding the user-defined allocated energy budget. Furthermore, this paper proposed a fast, low-cost heuristic that has optimal performance for the experimented problems sizes. The energy efficiency of the coarse-grained VFI-based system was compared to the optimal per-core DVFS on multiple benchmarks and with different energy budgets. While using multiple VFIs lowers the manufacturing and operating costs of manycore chips, the results showed that the VFI system's EDP, at the worst case, was within 13% of the EDP obtained by the per-core DVFS. The results also showed that the proposed framework gains greater EDP improvements for benchmarks with low CPU-intensive workloads.

According to [28], it is estimated that data centers in the U.S. are expected to consume electricity up to 73 billion kilowatt-hours per year from 2014 to 2020, which cost the American businesses $6 billion annually. Based on this report, the most efficient technologies and management practices will save energy up to 40% in 2020. Considering the system configuration used in this paper, the proposed framework saves more than 30% of energy in EB(L). Even if this framework reduces energy by 5% when it is deployed on larger system sizes, it will have a big economic impact on the energy costs of the future high performance computing.

REFERENCES

[1] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," *Proc. IEEE MICRO*, pp. 347-358, 2006.

[2] S. Borkar, "Thousand core chips: a technology perspective," *Proc. ACM DAC*, pp. 746-749, 2007.

[3] S. Pagani, A. Pathania, M. Shafique, J. J. Chen, J. Henkel, "Energy Efficiency for Clustered Heterogeneous Multicores," IEEE Trans. TPDS, pp. 1315-1330, 2017.

[4] S. Hajiamini, B. Shirazi, C. Cain, H. Dong, "Optimal Energy-Aware Scheduling in VFI-enabled Multicore Systems," *Proc. IEEE HPCC*, pp. 490-497, 2017.

[5] K. Greene. 2006. MIT Technology Review [online].

https://www.technologyreview.com/s/406760/the-trouble-with-multi-core-computers/ [retrived: July, 2018]

[6] W. Kim, M. S. Gupta, G. Y. Wei, D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," *Proc. IEEE HPCA*, pp. 123-134, 2008.

[7] U. Y. Ogras, R. Marculescu, D. Marculescu, E. G. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip," IEEE Trans. VLSI, 17, pp. 330-341, 2009.

[8] K. Duraisamy et al., "Energy efficient MapReduce with VFI-enabled multicore platforms," *Proc. ACM DAC*, pp. 1-6, 2015.

[9] U. Y. Ogras, R. Marculescu, D. Marculescu, "Variation-adaptive feedback control for networks-on-chip with multiple clock domains," *Proc. ACM/IEEE DAC*, pp. 614-619, 2008.

[10] R. David, P. Bogdan, R. Marculescu, U. Ogras, "Dynamic Power Management of Voltage-Frequency Island Partitioned Networks-on-Chip using Intel Single-Chip Cloud Computer," *Proc. ACM/IEEE NOCS*, pp. 257-258, 2011.

[11] S. Jin, S. Pei, Y. Han, H. Li, "A Cost-Effective Energy Optimization Framework of Multicore SoCs Based on Dynamically Reconfigurable Voltage-Frequency Islands," *ACM Trans. Des. Autom. Electron. Syst.* 21, pp. 1-14, 2012.

[12] M. Ozen and S. Tosun, "Genetic algorithm based NoC design with voltage/frequency islands," *Proc. IEEE AICT*, pp. 1-5, 2011.

[13] P. Ghosh and A. Sen, "Efficient mapping and voltage islanding technique for energy minimization in NoC under design constraints," *Proc. SAC*, pp. 535-541, 2010.

[14] S. Jin, Y. Han, S. Pei, "Statistical energy optimization on voltage–frequency island based MPSoCs in the presence of process variations," *Elsevier J. Microelectronics*, pp. 54, 23-31, 2016.

[15] N. Barrow-Williams, C. Fensch, S. Moore, "A communication characterisation of Splash-2 and Parsec," *Proc. IEEE IISWC*, pp. 86-97, 2009.

[16] Lap-Fai Leung and C-Y. Tsui, "Energy-aware Synthesis of Networks-on-chip Implemented with Voltage Islands," *Proc. DAC*, pp. 128-131, 2007.

[17] C. L. Chou and R. Marculescu, "Incremental run-time application mapping for homogeneous NoCs with multiple voltage levels," *CODES+ISSS*, pp. 161-166, 2007.

[18] M. A. Oxley et al., "Makespan and Energy Robust Stochastic Static Resource Allocation of a Bag-of-Tasks to a Heterogeneous Computing System," IEEE Trans. TPDS, pp. 26, 2791-2805, 2015.

[19] B. Hamann and J.L. Chen, "Data point selection for piecewise linear curve approximation," in *Computer Aided Geometric Design*, 11, pp. 289-301, 1994.

[20] N. Binkert et al., "The gem5 simulator," *ACM Comput. Archit. News*, 39, pp. 1-7, 2011.

[21] S. Li et al., "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," *IEEE Proc. MICRO*, pp. 469-480, 2009.

[22] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," *ACM Comput. Archit. News*, pp. 24-36, 1995.

[23] C. Bienia, S. Kumar, J. Singh, K. Li, "The PARSEC benchmark suite: characterization and architectural implications," *Proc. ACM PACT*, pp. 72-81, 2008.

[24] D. M. Gay, "The AMPL Modeling Language: An Aid to Formulating and Solving Optimization Problems," *Proc. Mathematics & Statistics*, pp. 134, 95-116, 2016.

[25] AMPL Products: Solvers. 2018. https://ampl.com/products/solvers/ [retrived: July, 2018]

[26] S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems," *IJCAET*, 6, pp. 440-459, 2014.

[27] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction," *Proc. IEEE MICRO*, pp. 81-92, 2003.

[28] https://eta.lbl.gov/publications/united-states-data-center-energy [retrived: July, 2018]