Thesis for The Degree of Doctor of Philosophy

Techniques to Improve Energy Efficiency on Heterogeneous Multiprocessors under Timing and Quality Constraints

Muhammad Waqar Azhar

Division of Computer Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2022

Techniques to Improve Energy Efficiency on Heterogeneous Multiprocessors under Timing and Quality Constraints

Muhammad Waqar Azhar

Thesis supervisor: Prof. Per Stenström, Chalmers University of Technology, Sweden

Thesis co-supervisors:
Dr. Miquel Pericàs, Chalmers University of Technology, Sweden
Dr. Vassilis Papaefstathiou, University of Crete/FORTH, Greece
Dr. Madhavan Manivannan, Chalmers University of Technology, Sweden

Examiner: Prof. Fredrik Dahlgren, Chalmers University of Technology, Sweden

Opponent: Prof. Josep Torrellas, University of Illinois, Urbana-Champaign, USA

Grading Committee:
Prof. David Black-Schaffer, Uppsala University, Sweden
Boris Grot, Associate Professor at University of Edinburgh, UK
Prof. Cristina Silvano, Politecnico di Milano, Italy
Prof. Philippas Tsigas (deputy member), Chalmers University of Technology, Sweden

Chairman: Prof. Pedro Trancoso, Chalmers University of Technology, Sweden

ISBN 978-91-7905-621-6
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 5087 .
ISSN 0346-718X

Technical Report No 211D
Department of Computer Science & Engineering
Division of Computer Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2022.

"I want to dedicate my work to my family, especially my Father, Mr. Muhammad Ramzan, who has always inspired me with his technical ability, supportive feedback, and futuristic ideas."

"the more you know, the more you realize you don't know"
- Aristotle

Abstract

Traditionally, applications are executed without the notion of a computational deadline and often use all available system resources which leads to higher energy consumption. User specification of Quality of Service (QoS) constraints, in terms of completion time and solution quality, opens up for allocation of *just enough resources* to an application to finish *just in time* and thereby save energy. Modern heterogeneous multiprocessor (HMP) platforms provide a set of configurable resources including a frequency range of dynamic voltage frequency scaling (DVFS), one among a set processor types, and one or a plurality of processors of each type. They can be configured at run-time to open up new opportunities for resource management.

This thesis presents techniques to reduce energy consumption under QoS constraints by allocating resources at run-time on heterogeneous multiprocessor platforms targeting sequential and parallel iterative and task-parallel applications. The proposed techniques rely on a progress-tracking framework that monitors and predicts how much time is left until the application finishes. Furthermore, the proposed framework enables the prediction of computation demand and performance requirements for future iterations or tasks.

The first contribution of this thesis is a resource management technique, called *SLOOP*, targeting single-threaded applications. *SLOOP* allocates resources, i.e., processor type and DVFS, for each iteration to meet deadlines while using the prediction of computational demand and execution-time.

The second contribution of this thesis is a resource-management scheme, called *SaC*, for multi-threaded applications executing on HMPs, where resources also include the number of processors besides DVFS and processor type. *SaC* first chooses the most energy-efficient configuration that meets the deadline. The proposed technique collects execution-time slack over subsequent iterations to select a configuration that can save energy.

The third contribution of this thesis is a resource manager, called *Task-RM*, for task-parallel applications executing on HMPs under QoS constraints. *Task-RM* exploits the variance in task execution times and imbalance between sibling tasks to allocate just enough resources in terms of DVFS and processor type. It uses an innovative off-line analysis to avoid redoing scheduling analysis at run-time.

Finally, the fourth contribution is a scheme, called *Approx-RM*, that can exploit accuracy-energy trade-offs in approximate iterative applications. *Approx-RM* allocates an appropriate amount of resources while guaranteeing timing and solution quality specifications. *Approx-RM* first predicts the iteration count required to meet the quality target and then allocates enough resources on an HMP in terms of DVFS, processor type and processor count to save energy while meeting a performance target.

Keywords

Energy Efficiency, Quality of Service, Resource management, Heterogeneous multiprocessor, Dynamic Voltage Frequency Scaling (DVFS), Core migration, Thread throttling, Soft real time systems.

Acknowledgment

I want to express my gratitude to my supervisor, Professor Per Stenström, whose invaluable guidance, practical suggestions, and constructive feedback shall be of benefit to me for the rest of my career.

I must also thank my co-supervisors, Dr. Vassilis Papaefstathiou, Dr. Miquel Pericàs, and Dr. Madhavan Manivannan, whose role has been instrumental in providing me with insightful suggestions and feedback that helped me polish my work further

I want to thank my examiner, Dr. Fredrik Dahlgren, for his beneficial and productive feedback and support for the completion of my Ph.D. studies.

I would also like to offer my special thanks to Professor Per Larsson-Edefors, with whom I began my journey as a researcher during my years of master's studies and whose unwavering guidance helped me broaden my perspective about research.

I also had the pleasure of working with my colleagues, Pedro, Lars, Jan, Ioannis, Risat, Evangelos, Petros, Ahsen, Prajith, Stavros, Albin, Stefano, Christoffer, Dimitry, Alexandra, Nadja, Bhavi, Monica, and others. I would particularly like to mention Mehrzad, with whom I had countless fruitful discussions pertaining to various research topics.

Finally, particularly helpful to me during this time were my parents, wife, siblings, and family, whose unwavering support has been instrumental to me throughout my course of Ph.D. I thank you all.

This research was supported by grants from the Swedish Research Council (grant no: 2012-4924, Dnr 2019-04929), Swedish Foundation for Strategic Research (Dnr CHI19-0048), the European Research Council (ERC) under the MECCA project (contract 340328), and European Union's Horizon 2020 research and innovation program (grant no: 826647)

List of Publications

Appended publications

This thesis is based on the following publications. The papers will be referred to in the thesis using their Roman numerals:

I. M. Waqar Azhar, Per Stenström, and Vassilis Papaefstathiou "SLOOP: QoS-Supervised Loop Execution to Reduce Energy on Heterogeneous Architectures."
In *ACM Transactions on Architecture and Code Optimization, Article No 41, Volume-14, Issue-4, Publication date: 05 December 2017.*

II. M Waqar Azhar, Miquel Pericàs, and Per Stenström "SaC: Exploiting Execution-Time Slack to Save Energy in Heterogeneous Multicore Systems."
In the *Proceedings of the 48th International Conference on Parallel Processing(ICPP 2019), Article No.: 26, Publication date: 05 August 2019.*

III. M Waqar Azhar, Miquel Pericàs, and Per Stenström "Task-RM: A resource manager for energy reduction in task-parallel applications under quality of service constraints."
In *ACM Transactions on Architecture and Code Optimization, Article No 11, Volume-19, Issue-1, Publication date: 23 January 2022.*

IV. M Waqar Azhar, Madhavan Manivannan, and Per Stenström "Approx-RM: Reducing Energy on Heterogeneous Multiprocessors under Accuracy and Timing Constraints."
To be submitted to IEEE Transactions on Computers.

Contents

Chapter 1

Introduction

1.1 Aim of the Thesis

Energy efficiency is one of the most crucial metrics in computer systems and its importance has been further accentuated with the end of Dennard Scaling [1, 2]. Energy efficiency is not only indispensable for battery powered systems, e.g., Internet of Things (IoT) devices, embedded systems, and smartphones, but also essential for server infrastructures such as data centers. Energy efficiency can enable operational longevity in battery-powered systems, while for servers, it can lead to lower cooling requirements and thereby lower electricity costs [3, 4].

Integrated circuits have enjoyed an exponential growth in terms of on-chip transistor count as a consequence of Moore's Law. However, single-threaded performance improvements have seen a slowdown due to higher power/energy consumption after the turn of the millennium leading to the multi-core era [5]. Moreover, the variety in application behavior such as compute/memory intensiveness, data-parallelism, instruction-level parallelism and memory-level parallelism calls for the integration of a variety of computing elements. Moreover, heat dissipation constraints rarely allow all of the chip-level computational units to be used at full throttle at the same time [6], thus enhancing the case for mixing energy and performance-oriented computational units (or processors). Coupled with the aforementioned facts and the availability of huge transistor counts, designers have decided to include general-purpose (GP) processors, graphical processing units (GPUs), and special-purpose processing elements, also called accelerators, in a single chip leading to *heterogeneous multi-processors* (HMP). There also exists heterogeneity within general-purpose processors, and it comes in two flavors. A first variant integrates in-order and out-of-order processors that are governed by the same instruction set architecture (ISA) e.g. ARM big.LITTLE [7]. A second variant employs identical processor architectures that are fabricated using different semiconductor cells, resulting in different performance/energy characteristics as in the Qualcomm Snapdragon [8]. This thesis considers heterogeneous multi-processors comprising different processor types with the same ISA.

As the behavior of workloads is evolving continuously, future requirements are hard to predict at design time. Thus, chip designers have left the management of modern parallel processing chips to the programmer/user. Difficulty in

managing complex transactions to achieve desired optimization goals concerning, e.g., performance, throughput or energy, has given rise to implementation of *resource management systems* as run-time systems. A *resource manager*(RM) hides the complexity of the underlying architecture from programmers by merely requiring high-level abstractions/specifications such that users may manage chip resources automatically. In the context of HMP, configurable resources include dynamic voltage frequency scaling (DVFS), selection of processor type and processor count. Moreover, in case of approximate applications, solution quality can also be used as a tuneable resource to save energy. This thesis focuses on resource management schemes that address these resource dimensions.

Traditional systems execute workloads without any notion of Quality of Service (QoS), typically using all available computational resources (e.g., the *performance* governor in Linux CPUFreq [9]). This QoS agnostic approach can overprovision resources and thus waste energy as certain applications need to produce results at a specific rate (throughput) or at fixed deadlines (completion deadlines) only. Executing faster than these deadlines do not add any value to the user. These deadlines are typically governed by real-world requirements, e.g., a specific video frame rate or program completion deadlines. Specification of quality requirements allows the resource manager (RM) to allocate *just enough* computational resources to an application or a phase of application to finish *just-in-time* before the deadline and thereby save energy.

In this context, this thesis aims to solve some of the challenges pertaining to energy efficiency by obtaining QoS specifications from users/programmers and designing *resource managers* that are able to allocate an appropriate amount of resources. The central theme here is to design resource managers that can take QoS specifications into account and predict the computational demand and execution behavior of workloads at run-time, in order to allocate befitting resources to save energy. The thesis targets heterogeneous multi-processors where resource management decisions include dynamic voltage frequency scaling (DVFS), selection of most appropriate processor type, and thread/processor count.

This thesis is based on four articles. The central theme is to save energy under QoS constraints for various types of applications executing on a heterogeneous multiprocessor platform. Each paper presents a resource manager designed for a specific application type operating under a unique set of constraints. **Paper-I** targets iterative sequential applications; **Paper II** focuses on multi-threaded iterative applications. **Paper-III** aims to improve energy efficiency for task-parallel applications, and finally, **Paper-IV** proposes a resource manager for approximate iterative applications.

1.2 State-of-the-Art

Conventionally, applications are executed without the notion of deadline or completion time requirements and typically use all or nearly all of the resources. Resource usage is only restricted by power and cooling constraints [9]. In this scenario, energy efficiency concerns are addressed by employing power gating [10–13]. On the other hand, *race-to-idle* techniques [14–17] are employed, where the processor is powered down to a sleep or idle state after applications finish. Here, it is essential to note that the duration of the idle period affects energy savings [3, 4]. Some techniques elongate the idle periods by employing methods like *Computational Sprinting* [6]; however, power consumption during the active phase will be higher due to the cubic relationship of energy to higher frequency and voltage. Moreover, *race-to-idle* schemes generate considerable execution time slack or (or simply *slack*) that can be defined as the difference between the deadline and execution times, and it can be used to slow down the execution and save energy.

The behavior of applications also exhibits a varying computational demand due to factors such as amount of instruction-level, memory-level or thread-level parallelism during the computational phase. Some proposals leverage these properties by using Dynamic Voltage Frequency Scaling (DVFS) [18, 19] or by selecting among one of many processor types [20] to save energy while keeping the same performance level. However, computational results are only required at a rate or instance specified by the user and offering more resources to run faster wastes energy. Thus, specification of QoS requirements allows managing resources while conforming to user specified performance constraints. Considering QoS and the vast configuration space offered by heterogeneous multi-processor platforms, there is a need for a holistic approach that can save energy by allocating computational resources at run-time depending on the computational demand. This is one critical gap in state-of-the-art that has been addressed in this thesis. The resource management challenges and solutions depend on factors such as application types (e.g., sequential or parallel), QoS specifications and programming models, therefore the following paragraphs shall present prior art for a few of these scenarios.

Considering sequential applications, Hughes et al. [21, 22] propose a domain-specific framework that leverages variability in execution time of processing video encoder/decoder frames and saves energy using allocation of DVFS and micro-architectural units (e.g., issue width, instruction window size and functional units). Suh et al. [23] present an approach that regulates performance against a specific target, i.e., the Instructions-Per-Second (IPS) rate, calculated offline using DVFS to save energy. Techniques in prior art show three main shortcomings: first, the methods are not application-agnostic; second, the computational demand is not estimated at run-time, but offline; last, only DVFS is considered and not other resources in HMPs such as processor types. This thesis addresses this gap by presenting application-agnostic methods that predict the computational demand at run-time to allocate just enough resources from the available configuration space.

Due to the availability of multi-cores, parallel applications can allow the resource manager to adapt the processor count along with the other available resources, such as DVFS and processor type, in the configuration space of HMPs.

However, a fundamental challenge that arises for on-line resource management is to predict application performance and energy behavior because of the large number of configurations available in HMPs with low overhead. Consequently, computational intensive machine learning (ML) models trained offline [24] or online [25] have been used in prior art for prediction. However, off-line training cannot fully capture the run-time behavior, while online training is associated with considerable overheads. In contrast, Li et al. [26] propose an online method that curtails the overheads by considering a small portion of the configuration space, pruned during the training phase. I observed three fundamental deficiencies in these previous works. First, the profiling-based training employing ML algorithms has considerable overheads. Second, the entire configuration space is not considered at run-time. Third, a simple slowdown policy, that uses the available slack as soon as it is available, limits the use of energy-efficient computational units, e.g., LITTLE cores, in HMPs. This thesis strives to address the deficiencies in the above-mentioned prior art.

Expression of parallel programs in a task-parallel model enables additional performance, but creates further challenges and opportunities in resource management. A task-parallel application can be modeled as a Directed Acyclic Graph (DAG) where nodes are tasks and arcs are dependencies among tasks. In this context, identifying energy-saving scenarios in such programs requires a scheduling analysis of a DAG using the execution time of tasks, requiring a considerable computational effort. Thus, off-line (static) techniques offer low run-time overheads, where off-line decisions employing DVFS [27, 28] or a combination of DVFS and processor type [29, 30] are used to save energy at run-time. However, off-line techniques assume off-line estimation of the execution time of tasks, which may lead to overprovisioning of resources and, consequently, more energy consumption. Kang et al. [31] present an online method that curbs the overheads by analyzing a small portion of the DAG at one instance, and only considering a single resource, i.e., DVFS. Moreover, this work assumes a fixed number of processors for the application's life span. In short, online proposals in prior art lack in three aspects. First, resource management algorithms are not suitable for use at run-time due to higher overheads. Second, the complete configuration space is not considered. Third, the schemes do not cater to the requirement of changing processor counts. This thesis provides a method to address the gap in the works mentioned above.

Finally, considering approximate iterative applications some proposals have focused on only providing quality guarantees [32, 33] without considering performance guarantees while using offline characterization. Several methods aim at maximizing solution accuracy under an energy budget without considering a performance constraint by employing offline application-specific resource estimation models [34] and application-specific accuracy controls [35]. Farrell et al. [36] propose a technique that employs application-specific controls to throttle accuracy for minimizing energy under timing constraints but provides no quality guarantees. Dayapule et al. [37] offer a method that uses offline estimations of performance and power to provide performance (i.e., tail latency) guarantee but no quality guarantees. Finally, Kulkarni et al. [38] employ application-specific offline characterizations to allocate resources, i.e., processor and memory allocation, to provide both quality and throughput guarantees. In short, prior art has several fundamental shortcomings: first,

the use of application-specific controls curtails the applicability beyond the considered workloads. Second, application-specific offline characterization is used to estimate the application's performance and energy, which restricts the full exploitation of potential of energy savings because the run-time behavior differ from the offline one. Finally, these techniques do not use the entire set of resources on offer, i.e., DVFS, processor type, and processor count, along with approximation. Overall, the existing proposals lack an application-agnostic method that reduces energy under accuracy and timing constraints through resource allocation based on online estimation of application performance and energy while using the entire configuration space offered by HMPs. This thesis presents a resource manager that addresses the shortcomings in proposals as mentioned above.

1.3 Problem Statement

This thesis presents a resource management framework aiming at reducing energy consumption for applications executing on heterogeneous multiprocessors (HMPs) under quality of service constraints. This raises the following questions that are addressed in the four papers (Paper I-IV) appended to the thesis:

1. The first problem is how to use a set of resources, specifically DVFS and processor type, at run-time to save energy by predicting the computation demand and execution time so as to finish a sequential application under the given performance constraint. This problem is addressed in **Paper-I**

2. The second problem is how to use execution time slack to save energy through resource allocation, i.e., DVFS, processor type, and processor count, based on an online prediction of the application's performance and energy behavior in multi-threaded applications executing on HMPs. This problem is addressed in **Paper-II**

3. The third problem is how to save energy through assigning resources, i.e., DVFS and processor type at run-time to tasks-parallel programs executing under QoS constraints on makespan on HMPs through the exploitation of variance and imbalance in task execution time for a range of processor count allocations. This problem is addressed in **Paper-III**

4. The fourth problem is how to exploit accuracy-energy trade-offs through allocation of resources in terms of DVFS, processor type and processor count to save energy in an approximate iterative applications while providing quality and timing assurances. This problem is addressed in **Paper-IV**

1.4 Thesis Contributions

This thesis makes four contributions

1. The first contribution of the thesis is a resource manager, called *SLOOP*, for iterative sequential applications executing on HMP platforms. *SLOOP* scheme first presents a progress-tracking framework based on monitoring of applications' outer loop. It uses the proposed framework to allocate *just enough* resources, as they apply to DVFS and processor type, by assessing applications progress in comparison to the deadline per iteration. *SLOOP* accomplishes this by employing a novel prediction mechanism that estimates future computational demand and application performance on the configuration space. These contributions are made in **Paper I**.

2. The second contribution is a novel slack management scheme – *SaC* (Slack as Currency)- that reduces energy by increasing the utilization of energy-efficient configurations in the system. *SaC* relies on an online prediction and search method that establishes application' performance and energy characteristics with low overhead and considerable accuracy. These contributions are made in **Paper II**.

3. The third contribution of this thesis is a run-time resource manager, *Task-RM*, for task-parallel applications to save energy under QoS constraints. *Task-RM* leverages an innovative offline analysis scheme that defines soft deadlines of the tasks within the program for a variety of scenarios, thus avoiding program analysis under changing hardware scenarios. *Task-RM* uses an efficient performance and energy prediction mechanism to assist in resource allocation. These contributions are made in **Paper III**.

4. The fourth contribution is *Approx-RM*, a resource allocation scheme for approximate iterative applications executing on HMPs that saves energy under accuracy and timing constraints by means of controlling DVFS, processor type, and processor count. *Approx-RM* offers a lightweight mechanism based on curve fitting to predict application duration in terms of number of iterations. Furthermore, a performance and energy prediction mechanism is proposed to estimate application behavior spanning the HMP resource allocation space. These contributions are made in **Paper IV**.

Chapter 2

Summary of the Papers

2.1 Summary of Paper I

Traditionally, *race-to-idle* is used to save energy, where the application executes using the available resources and then switches to a power-down state (or idle) [14–17]. *Race-to-idle* techniques over-provision the resources and result in high energy consumption. Applications also show varying behavior in instruction-level parallelism (ILP) and memory-level parallelism (MLP), which affect the computational demand. Thus, some proposals employ DVFS [18, 19] or processor type selection [20] to reduce the stall cycles while providing the same performance as *race-to-idle*. Limited availability of such variations, e.g., ILP and MLP, result in small energy savings.

Introduction of QoS constraints can unlock further opportunities, where an allocation of an appropriate amount of resources, that in context of HMP include DVFS and processor type, to applications enable *just-in-time* completion before the deadline and save energy. In this regard, Hughes et al. [22] propose a domain-specific proposal that adjusts DVFS and micro-architectural configuration (i.e., issue width, instruction window size, and functional units) to meet a specific video frame rate by characterizing the frame type. Kluge et al. [39] present a similar solution while only using DVFS to regulate the execution speed in video encoding/decoding applications. These techniques are based on specific application insights and are not application-agnostic. Sue et al. [23] propose to regulate instructions per second (IPS) to a predefined level based on average or worst-case IPS measured offline to improve energy efficiency. Assigning resources based on a constant computational demand, i.e., IPS rate, that is computed offline, can lead to over-provisioning and consequently higher energy consumption.

The prior art mentioned above has a few fundamental drawbacks. Firsts, the application computational demand is estimated offline. Second, some proposals rely on application-specific information, i.e., frame types. Lastly, the schemes do not use the entire configuration space offered in HMPs.

Paper-I targets the problem of resource, i.e., DVFS and processor type, allocation to save energy by predicting the computation demand and execution time at run-time to finish close to the deadline given by QoS specification. It proposes a framework, called SLOOP, for resource management for iterative sequential applications executing on HMPs under QoS constraints expressed as a deadline per iteration (or throughput). A streaming application model is assumed, where an outer loop encapsulates the computational kernel and every iteration operates on new input data.

SLOOP proposes to use loop iterations to monitor applications progress and allocates resources in connection with the deadline to allocate an appropriate amount of resources in terms of DVFS and processor type to subsequent iterations. Since every iteration finishes before its deadline, a slight execution time slack (or simply *slack*) is generated. Slack is defined as the difference between the deadline and the execution time, and it can be used to slow down future iterations. In this context, it is assumed that data for each iteration is available well before its start.

Estimating computational demand and application performance on the set of processors in an HMP are two fundamental sub-problems. As for the first requirement, the instruction count from previous iterations is stored and

used to predict the instruction count for future iterations using two types of predictors: an average predictor and a gradient predictor. The application execution time on various processor types is estimated by storing the cycle per instruction (CPI) count of already executed iterations and using it to predict CPI for future iterations. CPI is stored for all the processor types available in the HMP. The predicted instruction count and CPI can be used to predict the minimum frequency that can satisfy the deadline per iteration requirement for each type of processor.

Different processor types, e.g., big, LITTLE [7, 40] in HMPs have different performance levels and can fulfill the given QoS requirement at different voltage-frequency (V-F) settings and lower DVFS states, offering more energy savings. *SLOOP* uses this insight to predict the minimum DVFS setting that can meet the deadline for all processor types in the HMP and then uses the most energy-efficient processor type and DVFS setting. Since the LITTLE cores offer lower energy, they are preferred, provided the predicted frequency lies within the legal limit. This process of prediction and resource allocation repeats after each iteration. Eventually, each iteration finishes before the deadline. In a nutshell, the resource allocation is adjusted to the dynamic computational demand.

The SLOOP framework is evaluated on an ODROID-XU3 board containing Samsung Exynos-5422 chipset [41] that is based on ARM's big.LITTLE technology using iterative applications from the ALPBench [42] and SPEC2006 [43] benchmark suites. The deadline is defined using *Race-to-idle* that executes the workload at the highest resource setting, i.e., a 2-GHz big core, and then idles. The deadline is set equal to the execution time of the slowest iteration among all the application main-loop iterations. This deadline setting ensures that all the iterations meet the deadline while executing at the highest resource allocation.

SLOOP is evaluated against *Race-to-idle* as the baseline, the scheme proposed in state-of-the-art by Sue et al. [23] (henceforth referred to as *DMIPS*) and an *oracle* scheme. Here, *Oracle* refers to a scheme that makes a perfect prediction and uses SLOOP's resource manager. *DMIPS* has two variants: the average profile and worst-case profile, where the former uses the average instruction count and the latter uses the worst-case instruction count as prediction and these values are computed using the instruction count per iteration trace of baseline. Both schemes use SLOOP's resource manager.

SLOOP saves 25% energy compared to *Race-to-idle* and is only 8% worse than *Oracle*, without missing any deadlines. *DMIPS worst-case profile* performs the same as *Race-to-idle* and does not save any energy while *DMIPS average profile* saves 24% energy compared to *Race-to-idle* but misses 97% of the deadlines. Moreover, SLOOP only incurs 0.06% and 0.07% of timing and energy overheads, respectively. By further relaxing the deadline by 20%, 50% and 100%, the energy savings increase by 42%, 52%, and 63%, respectively.

2.2 Summary of Paper II

The saturation of single-threaded performance improvements and the availability of many transistors on modern semiconductor chips have enabled the transition into the multi-core era. Moreover, various application types require different hardware properties forcing the designers to package various computational units, i.e., in-order and out-of-order processors, graphical processor units, and custom accelerators in a single chip. Such a setting is beneficial in exploiting the performance and essential for keeping power or energy in check. This paper focuses on multi-threaded programs executing under a quality of service specification regarding the computational deadline on single-ISA heterogeneous multiprocessors. The aim here is to design a resource manager that can save energy by employing resource allocation concerning DVFS settings, processor type selection, and processor count.

In this context, *Race-to-idle* [14–17] over-provisions the resources; thus, it is not suitable. In contrast, the introduction of QoS requirements calls for the allocation of *just enough* resources that are sufficient to meet the computational deadlines. Therefore, several techniques [24–26] propose to slow down the execution to finish just before the deadline and save energy. However, the prediction of applications' execution time and energy consumption across the configuration space is not trivial, resulting in some techniques [24, 25] employing computationally intensive machine learning models for prediction. In this context, offline training-based approaches [24] yield low accuracy when exposed to unknown scenarios at run-time, whereas in the case of online training [25] the overheads become a concern. In contrast, Li et al. [26] proposes to limit the overheads by providing a heuristic that only considers a small portion of the configuration space.

The shortcoming in the prior art can be summarized as follows. First, application timing and energy behavior are either estimated offline or using computationally intensive methods at run-time. Second, the entire configuration space is not used for resource allocation. Last, the schemes use the slack as it is produced, limiting the use of energy-efficient configurations such as LITTLE cores.

This paper targets the problem of saving energy in iterative parallel applications. It does so by devising a novel slack management scheme that allocates resources so that it accumulates and consumes slack to increase the utilization of energy-efficient configurations. Furthermore, we approach the problem of designing an online low overhead prediction method that can estimate applications' performance and energy behavior to enable the above-mentioned resource allocation.

Thus, **Paper II** proposes a resource management policy, referred to as *SaC* - Slack as Currency- that saves energy under a constraint on application's completion time. This paper further proposes a lightweight online prediction method to enable resource allocation. Similar to **Paper I**, this paper also uses the iterations of the application's outer loop to monitor it's behavior and allocate it on a new configuration.

QoS specification is defined in terms of the deadline for the program. The outer loop of the application is used to monitor, progress tracking, and predict application performance and energy. Assuming that application iteration-count

is given, the soft deadlines for each iteration are computed at the start of execution. Comparing the execution time at the end of each iteration with soft deadline enables *SaC* to calculate slack and ascertain if the application is executing faster or slower than required. If an iteration completes early, then it results in positive slack that can be used to save energy since it is only necessary to complete before the application deadline; individual iteration can execute slower or faster without violating overall QoS specification.

In this context, *SaC* first selects an energy-efficient configuration, called *slack generating configuration (SGC)*, which meets the soft deadline per iteration target. Since the timing behavior of the configuration space is quantified, there is always some tiny slack. Then slack is allowed to build up across multiple iterations (or accumulated) by keeping the soft deadline for all iterations the same. As slack builds up and reaches a threshold, the RM uses slack to relax the soft deadline per iteration and chooses an energy-efficient configuration, called the *slack using configuration* (SUC), that would typically violate the soft deadline per iteration. Since it is desirable to execute multiple iterations in *slack using (SU) mode*, the accumulated slack is distributed over several iterations. Once the slack approaches zero, a new SGC is selected, and this process repeats over and over again.

The method of selecting the SGC and SUC is based on the fact that the minimum voltage frequency that meets the deadline yields minimum energy for a specific thread count and processor type. So the minimum frequency for all the combinations of processor types and thread counts is computed. Then, the configuration with the minimum predicted energy is chosen. Next, the selection of a SUC is done with the same procedure but by relaxing the deadline by the amount of accumulated slack. However, the method of predicting the energy is different in the case of SUC. This is because different SUCs consume the accumulated slack in a different number of iterations. Thus, a configuration with lower energy per instruction (EPI) but smaller iterations in SU mode will have a lesser impact on energy efficiency than a configuration with slightly higher EPI and more iterations in SU mode. In short, the iteration count for all the SUCs is predicted, leading to the prediction of the average EPI. Finally, the configuration with the smallest predicted average EPI is selected as the SUC.

The performance and energy prediction method consist of two phases, a profiling-based training and a steady-state. During the training phase, the application is executed at pre-defined configurations and instructions per second (IPS) and EPI is measured and recorded in a table. The first two iterations are executed using maximum thread count at the maximum and minimum frequencies. The third and fourth iterations are then executed using the minimum thread count at maximum and minimum frequencies, respectively. Since each processor type in the system has a specific performance and energy characteristic, the same training procedure is repeated for each processor type. Then, the IPS and EPI are computed using interpolation at frequencies between the minimum and maximum frequency.

During steady-state, the resource manager measures execution statistics, including instruction count, cycle count, and energy consumption, at the end of every iteration and hence finds a suitable configuration to execute the next iteration. A history of the instruction count from previous iterations is

maintained and used for predicting the instruction count for future iterations, referred to as workload prediction. The cycle count is used to compute the execution time and slack. The resource manager keeps track of the accumulated slack.

SaC is evaluated on an Odroid XU3 platform using applications from the Rodinia [44] benchmark suite. First, two oracle schemes, *SaC optimal* and *Static optimal* are used as references of upper bounds, where both schemes have future knowledge and employ exhaustive search in the configuration space. *SaC optimal* finds SGC and SUC pairs and the points where to switch between SGC and SUC, while the *static optimal* finds the best single configuration that meets the deadline with minimum energy. The deadline used in this evaluation is set to the fastest configuration using LITTLE cores, i.e., 4 threads and a 1.4 GHz clock frequency. The deadline is set equal to $0.7 \times$ **slowest iteration** at this configuration. This deadline restricts the usage of LITTLE cores in normal scenarios.

SaC optimal saves 10% more energy compared to *Static optimal* using Race-to-idle as a baseline. This experiment shows the additional potential of energy savings that can be exploited by using slack. Another way of analyzing the behavior of both schemes is to look at the usage of energy-efficient configurations, for example, those comprising of LITTLE cores. *SaC optimal* and *Static optimal* use the LITTLE cores for 43% and 10% of iterations, respectively. This shows that *SaC optimal* is capable of exposing the most energy-efficient configurations that do not meet the deadline without exploiting slack.

Next, *SaC* is compared with *SaC optimal* and Li, according to Li et al. [26]). *SaC* saves 62% and 27% more energy compared to *Race-to-idle* and *Li*, respectively. Moreover, *SaC* is only 8% worse off than *SaC optimal*. The energy savings of SaC can be divided into two parts. First, a considerable amount of energy savings (i.e. 48%) come from selecting a suitable SGC. Second, further energy savings (i.e 14%) are achieved by using slack to expose the energy-efficient configurations in SU mode.

2.3 Summary of Paper III

Parallel applications can be generally represented as direct-acyclic graphs (DAG), where nodes are tasks and edges are dependencies between tasks. Such a representation enhances the possibility of harnessing performance, especially in the case of inherently sequential algorithms by employing the methodology of pipeline parallelism [45]. *Paper III* aims to reduce energy consumption under QoS constraint on the completion time for a task-parallel application executing on a single Instruction Set Architecture (ISA) heterogeneous multi-core platform (HMP), e.g., ARM *big.LITTLE* [7].

In the context of a DAG, QoS constraints are expressed as the worst-case schedule length (WCSL), and methods from prior art [27, 29, 31] are used to establish it. An offline simulation of a DAG's execution by assuming each task's upper-bound execution time (UBET) results in a worst-case schedule length (WCSL or makespan). Setting the program's deadline equal to or greater than the WCSL ensures its accomplishment, even if all the tasks take a time that amounts to UBET to finish. The UBET can be established by using measurement-based deterministic timing analyses (MBDTA) [46], and the DAG can be generated during compilation.

Specification of a program (DAG) deadline allows the resource manager to allocate appropriate resources, i.e., processor type and DVFS, to finish the application close to the deadline, thereby saving energy. In turn, this resource allocation requires that the task soft deadlines or latest finish time (LFT) must be estimated. Given the task LFTs, the resource manager can manage the resource allocation at run-time to save energy.

There are three main avenues for saving energy by slowing down a task execution while meeting the program deadline: First, tasks can finish early compared to its UBETs. The second scenario is the imbalance between a task's predecessors, where one or more tasks finish earlier than others. Since a successor task can only start after all of the predecessors have been completed, the predecessor tasks completed earlier than required can be slowed down to save energy. The third opportunity is enabled by the availability of additional processors. The user decides to change the hardware platform to one with additional processors, or the operating system momentarily allocates additional resources are some of the scenarios where it can happen. All these situations allow the RM to slow down tasks while adhering to the program deadline.

Prior art can be generally divided into two categories: offline (static allocation) or on-line (dynamic allocation) techniques. Offline techniques offer low run-time overhead and typically use computationally intensive optimization algorithms. Baskiyar et al. [27] and Alonso et al. [28] provide two techniques that identify the critical path and slow down the tasks that do not lie on it by using DVFS. Kumar et al. [29] use a user-specified extension of the WCSL as the deadline to save energy by means of DVFS and processor mapping. Lee et al. [30] propose a scheduling method that minimizes the computational energy by V-F scaling and reduces the communication energy using processor mapping. All these techniques use offline estimations of the UBET of each task; thus, they neither cater to the scenario of a task finishing earlier nor do they address the situation of additional processors, leading to over-provisioning of resources and, consequently, more energy consumption.

The on-line or dynamic techniques make decisions at run-time based on the behavior of the task-parallel application. Kang et al. [31] propose an on-line method based on progress tracking of the application to identify each task's early-finish time and slow down the subsequent tasks using DVFS. This method suffers from high overheads as the computationally intensive static scheduling algorithm is run every time a task finishes early or late – furthermore, neither processor mapping nor additional processors are considered limiting the energy-saving potential. To fully harness the energy-saving potential, all the energy-saving opportunities must be evaluated at run time with negligible overhead.

Paper III targets the problem of saving energy using run-time resource assignment, i.e., DVFS and processor type to tasks-parallel programs under QoS constraints on makespan by exploiting variance and imbalance in task execution time for a variety of processor allocation scenarios. **Paper III** addresses the shortcomings in the prior art by proposing a novel resource management approach, *Task-RM*, that consists of two steps. The first step is *offline analysis* that sets the soft deadline for each task in a DAG. Here, the task deadlines are computed for a range of processor counts solving the challenge of portability to new hardware platforms. The run-time resource manager tries to allocate the resources to meet the task deadlines. Adhering to task deadlines ensures that the program finishes close to the deadline.

We advocate that the process of task-scheduling must be separate from resource management decisions, such as processor type and DVFS. Consequently, our scheme can be used with any state-of-the-art scheduling method. The offline analysis employs the user-defined task-ordering criteria and QoS specification to compute the latest finish times (LFT) of the tasks for a range of processor counts using the UBETs of the tasks. The offline analysis consists of two steps: an offline simulation of a schedule using task UBETs, scheduling method, and processor count resulting in task execution schedule. This step is repeated for a range of processor counts resulting in a set of schedules. As a second step, each schedule is analyzed where, among other things, task imbalance is analyzed, and LFTs are adjusted. Secondly, for the case of additional processors, the LFTs of tasks are increased as much as possible while adhering to the program deadline. Finally, this information is compiled in a so-called *Latest Finish Time* table (i.e., LFT-table).

The resource manager uses the LFTs of the tasks – for the available processor count – as the soft deadline to allocate enough resources to each task at run-time to save energy, thus avoiding recompilation of the schedule for the new processor count. To assign the resources, that RM must carry out two sets of predictions. First, it needs an estimation of the task-computation demand that is accomplished by storing the history of the instruction count of tasks and using averaging to predict the instruction count for future tasks. Secondly, task performance and energy estimation are required to make resource allocation decisions. In this regard, a prediction mechanism is devised based on storing the Cycle Per Instruction (CPI), Misses Per Kilo Instructions (MPKI), and Effective capacitance (C_{Eff}) for the executed task to predict the future task using averaging. The resource allocation algorithm predicts the minimum frequency that can meet the deadline for each processor type in the system. Then the energy consumption at the predicted frequency is estimated. Finally,

the processor expected to have minimum energy is allocated the task.

Task-RM is evaluated on an assumed sixteen-core platform with eight big and eight LITTLE cores arranged in four clusters. The trace-based simulation using real execution and energy statistics measured on an ARM big.LITTLE platform (ODROID XU-3 board with Exonys 5422 [47]) is used to compile the results. The workloads [48] used are adopted from the BSC Application Repository (BAR) [49]. Four schemes, i.e., Race-to-idle, Dynamic Slack Allocation (DSA) [31], Oracle and Task-RM, are used in the evaluation. The program deadlines are set equal to WCSL assuming the baseline processor allocation i.e. 2-big cores, 2-GHz.

First, we analyzed the energy savings for a fixed processor allocation where *Oracle* and *Task-RM* offer an average energy savings of 36.66% and 33.55%, respectively with respect to *Race-to-Idle*. The energy savings for *Oracle* provide the upper limit of possible energy savings. It is also important to note that *Task-RM* performs close to *Oracle* with only 8% fewer energy savings, thus demonstrating its effectiveness in harnessing energy savings. Furthermore, *DSA* delivers an energy saving of 15.11%. In other words, *Task-RM* shows approximately 22% more energy savings compared to *DSA*.

Next, the energy savings are analyzed when additional processors are allocated compared to a baseline allocation. The energy savings increase as more processors are allocated. Still, they saturate at an allocation of eight big and four LITTLE processors with energy savings of 58.8% for Oracle and 55.6% for Task-RM, respectively. The availability of additional cores only increases energy savings if there are ready tasks to execute (i.e., sufficient parallelism in the DAG). The energy and timing predictions used in resource allocation decision-making show a considerable high accuracy with an average of 95% and 93% for timing and energy predictions, respectively. Lastly, *Task-RM* incurs less than 1% of timing and energy overheads, making it highly suitable for run-time use.

2.4 Summary of Paper IV

Approximate iterative applications (AIA), e.g., iterative solvers, are characterized by the fact that the *solution quality* improves with each iteration and applications finish when the *solution error* reaches a user-defined target. However, improvement in the solution quality saturates as the execution proceeds. Thus, a slight relaxation in the solution error target can reduce the iteration count to reach the solution quality target. This enables the resource manager to reduce the resource allocation resulting in a significant energy reduction. **Paper IV** targets the energy reduction for AIA on heterogeneous multicore platforms, e.g., ARM *big.LITTLE* [7] while providing the statistical guarantees on QoS constraints for timing and accuracy through appropriate resources allocations, i.e., processor type, processor count, and voltage-frequency (V-F) settings.

A considerable number of early proposals [23,26,50–54] have not considered accuracy and only used resource allocation in terms of DVFS, processor type, and processor count for improving energy efficiency under QoS constraints on performance. However, the trade-off between computation accuracy and energy reduction has been increasingly studied recently. Zhang et al. [32] target energy reduction under accuracy constraints by providing schemes for iterative methods and artificial neural networks (ANN) [33]. However, both these works do not consider timing requirements and use offline characterization. Vassiliadis et al. [34], and Hoffmann et al. [35] provide methods for maximizing accuracy under an energy budget but do not provide timing guarantees. Moreover, they use an offline-trained application-specific model [34] or application-specific approximation controls [35]. The technique from Farrell et al. [36] provides timing guarantees while reducing energy. However, it does not only fail to provide quality guarantees but uses application-specific accuracy controls. Dayapule et al. [37] propose guaranteeing the tail latency requirements and curtailing the accuracy degradation but do not provide the quality guarantees and use offline power and performance estimations. The technique proposed by Kulkarni et al. [38] provides both the quality and throughput guarantees by using dynamic recompilation to modify the processor and memory allocation but uses offline characterization of the approximation design space for each application.

In short, prior art has several fundamental shortcomings. First, the use of application-specific approximation control limits the service beyond the applications employed. Second, application-specific offline analysis or characterization curtails the energy-saving potential as run-time scenarios differ from offline ones. Collectively, existing proposals do not offer a method to reduce energy under both accuracy and timing constraints in an application-agnostic manner that do not rely on application-specific offline analysis. Moreover, no technique uses the entire configuration space, i.e., DVFS, processor type, and processor count, along with approximation as a means to reduce energy consumption.

Paper IV proposes an application-agnostic framework for reducing energy consumption when running approximate iterative applications on heterogeneous multiprocessors. The approach taken is to trade a slight yet controlled accuracy loss for energy while meeting timing deadlines. **Paper IV** presents *Approx-RM*. *Approx-RM* allocates just precise resources in terms of DVFS, processor type,

and processor count to meet the timing and accuracy constraints.

First, *Approx-RM* predicts the application iteration count to reach the relaxed solution error target (accuracy constraint) and then uses it to estimate a soft timing deadline for each iteration. Then application performance and energy behavior on various configurations of the HMP is predicted. Finally, the configuration estimated to meet the timing constraint and have minimum energy is chosen. This process continues until the application concludes.

A solution based on curve fitting is proposed for the prediction of the duration of the applications. The old samples of the solution error are stored in a first in first out (FIFO) buffer and at the end of every iteration, a new sample is pushed into said buffer and the oldest sample is discarded. Curve fitting is done once the buffer contains all the new samples, thus curtailing the overheads. The duration prediction first employs curve fitting to find parameters to a decaying exponential mode, i.e., slope and intercept, and then uses it to predict the duration of the applications in terms of iteration count required to reach the required solution quality target.

The predicted iteration count, application deadline, and executed iteration count are used to establish the soft deadline per iteration for the remaining iterations. Finally, the soft deadline per iteration and performance and energy prediction are used to find a minimum energy configuration that meets the deadline per iteration requirement.

The performance and energy prediction mechanisms are based on storing the history of instruction count, base cycle per instructions (CPI_0), misses per kilo instructions ($\mathrm{MPKI}_{\mathrm{LLC}}$) for the last level cache (LLC) and effective capacitance (C_{eff}) by reading hardware performance counters and on-board energy sensors after each iteration. In this context, a history table is maintained, and values are stored as per processor-thread combination. The future values are predicted using averaging of valid samples in the history buffer.

The resource allocation algorithm first uses the prediction model to predict the minimum frequency for all the combinations of processor type and processor counts. It then picks the minimum energy configuration while meeting the timing requirements. This process is repeated after each iteration and continues until the conclusion of the application. The reason is that the prediction of the duration and timing and energy behavior of the application evolves throughout application execution. Therefore, *Approx-RM* re-evaluates the duration and architectural behavior to re-allocate the resources. All of these mechanisms impose low run-time overheads.

Approx-RM is evaluated on an ARM big.LITTLE platform (ODROID XU-3 board with Exonys 5422 [47]) containing four big and four LITTLE cores organized in two clusters. Two sets of applications (published on web [55]) are used, first applications from BSC Application Repository (BAR) [49] modified to OpenMP and second specifically developed microkernels for multi-variate linear regression (implemented in C++ using OpenMP).

Approx-RM is compared against *Race-to-idle* and *Oracle*. *Race-to-idle* executes the applications at the fastest configuration, i.e., big-core, four threads and 2 GHz, and then powers down. In contrast, *Oracle* allocates the resources based on timing requirements but has perfect knowledge of the duration, performance and energy behavior on the hardware of the applications. This scheme serves as an upper bound of the energy savings on offer.

First, we analyze the energy savings by relaxing the solution quality target. *Oracle* and *Approx-RM* show an average energy saving of 23% and 11%, respectively, compared to *Race-to-idle* with no reduction in solution quality target and 35% and 26% energy savings, respectively, with 1% increase in solution quality target. The additional energy savings with relaxation in quality target is due to a reduction in iteration count by 11%. The reduction in iteration count allows the increase in time allocation per iteration, thus allowing the resource manager to reduce resources and save energy. Moreover, *Approx-RM* is effective in harnessing the energy savings as it performs close to *Oracle*.

The *Oracle* and *Approx-RM* shows the slack, that is, the difference between the deadline and the execution time, of 4% and 15% with respect to the deadline. This means that *Oracle* finishes very close to the deadline, and *Approx-RM* finishes somewhat earlier compared to *Oracle*. This is also an indication of the energy difference between the two schemes, as *Oracle* is more effective in utilizing the slack. The timing and energy prediction offers at least 93% and 94% accuracy, respectively. Lastly, the timing and energy overheads of *Approx-RM* scheme are approximately less than or equal to 1%.

Chapter 3

Concluding Remarks and Future Work

This thesis investigates the problem of energy reduction through appropriate resource management under QoS constraints for applications executing on heterogeneous multiprocessor (HMP) platforms. Specification of QoS in terms of computational deadlines and solution quality allows the resource manager to allocate *just enough* resources to fulfill the QoS requirements and thus save energy. The fundamental aspect here is that the user-specified QoS requirements enable the resource manager to maintain a lower performance and trade it for energy savings. This thesis targets single- and multi-threaded applications of various types, i.e., data-parallel, task-parallel, and iterative approximate applications. Furthermore, it takes advantage of a multi-dimensional configuration space, i.e., dynamic voltage/frequency scalaing (DVFS), processor type, and processor count, offered by HMPs. The focus of the thesis is to design low overhead resource management techniques that can be employed at run-time to save energy under QoS constraints.

The first contribution of the thesis (detailed in *Paper I*) is a scheme, i.e., *SLOOP* for resource allocation, in terms of DVFS and processor type, to monitor applications at the granularity of loop iterations and predicting future computational demand and execution time to finish on time. Evaluation of *SLOOP* has demonstrated that considerable energy can be saved by such a resource allocation policy while conforming to QoS specifications and incurring negligible overheads.

The second contribution of the thesis (detailed in *Paper II*) is a resource management scheme, called *SaC*, for multi-threaded applications, where the resources not only include the DVFS and processor type but also processor count. *SaC* first selects an energy-efficient resources allocation that meets the deadline. Then it uses the accumulated slack to utilize energy-efficient resources, e.g., low-power cores or low frequency on big cores, that generally violate deadline requirements. This methodology unlocks additional energy savings. Additionally, *SaC* also proposes a low-overhead technique for online prediction of performance and energy expenditure for applications on the entire configuration-space based on interpolation.

As the third contribution, this paper presents *Task-RM*, a scheme to save

energy in task-parallel applications executing on HMPs under QoS constraint on finish time (also known as makespan). *Task-RM* identifies the variance in execution time by monitoring each task and using the appropriate resource allocation in terms of DVFS and processor type to save energy. The performance requirements of tasks are established through an offline analysis that computes soft deadline of tasks for a set of processor counts enabling the *Task-RM* to save energy at run-time for a range of processor allocations without the need for re-analysis. The run-time resource manager uses the soft deadlines of tasks and performance and energy predictions to allocate an appropriate amount of resources to the tasks to finish close to their soft deadline, thus ensuring the completion time of parallel programs close to their deadlines.

Finally, the fourth contribution of this thesis is *Approx-RM* that exploits the accuracy-energy trade-off in approximate iterative applications. *Approx-RM* saves energy employing appropriate resource allocations while providing the solution quality and timing guarantees. *Approx-RM* first predict the application duration in terms of iteration count required to reach the user-specified solution quality target by applying curve-fitting on recorded samples of the solution error. *Approx-RM* then allocates appropriate resources in terms of DVFS, processor type, and processor count using performance and energy predictions to finish before the deadline to save energy.

Concerning future work, there are several avenues. First, the insights from this thesis can be used to extend the resource management framework to server-based applications where the QoS requirements are different, and application characteristics offer unique opportunities to save energy. Server applications typically consist of tasks invoked repeatedly from external events (from the user) with timing constraints on tail latency. The repeated invocation of jobs opens up opportunities to understand application behavior at runtime; however, non-availability of slack (in a traditional viewpoint) due to strict tail latency provision requires a novel resource management policy.

Secondly, we would like to apply the relevant findings of this work on graphical processing units (GPUs) that offer a different set of challenges and opportunities where the historical focus is on performance. Most existing proposals do not employ QoS and propose a methodology similar to race-to-idle [56]. QoS specifications allow the resource manager to save energy through resource allocation, e.g., compute cores, DVFS, and others. Considering the variety of application domains, e.g., graphics, machine learning (ML), and scientific computing (SC) benefiting from GPUs, the first problem is identifying and reasoning about QoS metrics. In some cases, i.e., graphics, it is well established, but resource allocation proposals fail to utilize it. In other instances, i.e., ML and SC, the QoS specification is not explicitly available, thus requiring a thorough analysis of use cases. In this context, the CPU-based resource management techniques cannot be applied to GPUs, and there is a need to develop novel solutions.

Lastly, emerging applications, e.g., data analytics, machine learning and genome analysis, are finding their way into embedded or edge devices. These applications have high computational demand and often operate on a large amount of data putting pressure on both computation units and memory systems. On the other hand, embedded devices have limited compute and memory capabilities and stricter power/energy constraints. Hence, traditional

architectures hamper the adoption of such applications. Therefore, identifying and adopting QoS specifications can stimulate numerous new design opportunities that are energy efficient, while delivering the required performance. Furthermore, since these applications are often executed on domain-specific accelerators, such as GPUs, the problem dimension and possibilities further expand.

Bibliography

[1] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, and A. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 10 1974.

[2] M. Bohr, "A 30 Year Retrospective on Dennard's MOSFET Scaling Paper," *IEEE Solid-State Circuits Newsletter*, vol. 12, no. 1, pp. 11–13, 2009.

[3] S. Kaxiras and M. Martonosi, *Computer Architecture Techniques for Power-Efficiency*, 2008.

[4] M. Själander, M. Martonosi, and S. Kaxiras, *Power-Efficient Computer Architectures: Recent Advances.* Morgan & Claypool Publishers, 12 2014, vol. 9, no. 3.

[5] K. Olukotun, L. Hammond, and J. Laudon, *Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency.* Morgan & Claypool Publishers, 2007, vol. 2, no. 1.

[6] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin, "Computational sprinting," in *IEEE International Symposium on High-Performance Comp Architecture*, Feb 2012, pp. 1–12.

[7] I. Lin, B. Jeff, and I. Rickard, "Arm platform for performance and power efficiency — hardware and software perspectives," in *2016 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, April 2016, pp. 1–5.

[8] A. Cabrera, S. Hitefield, J. Kim, S. Lee, N. R. Miniskar, and J. S. Vetter, "Toward performance portable programming for heterogeneous systems on a chip: A case study with qualcomm snapdragon soc," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021, pp. 1–7.

[9] D. Brodowski, N. Golde, R. J. Wysocki, and V. Kumar, "Cpu frequency and voltage scaling code in the linux (tm) kernel," *Linux kernel documentation*, p. 66, 2013.

[10] M. Kondo, H. Kobyashi, R. Sakamoto, M. Wada, J. Tsukamoto, M. Namiki, W. Wang, H. Amano, K. Matsunaga, M. Kudo, K. Usami, T. Komoda, and H. Nakamura, "Design and evaluation of fine-grained power-gating for embedded microprocessors," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 2014, pp. 1–6.

[11] Hailin Jiang, M. Marek-sadowska, and S. Nassif, "Benefits and Costs of Power-Gating Technique Abstract," *2005 International Conference on Computer Design*, pp. 559–566, 2005.

[12] K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka, "Power gating with multiple sleep modes," in *7th International Symposium on Quality Electronic Design (ISQED'06)*, 2006, pp. 5 pp.–637.

[13] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units," *Proceedings of the International Symposium on Low Power Electronics and Design*, vol. 2004-Janua, no. January, pp. 32–37, 2004.

[14] S. Dawson-Haggerty, A. Krioukov, and D. E. Culler, "Power optimization – a reality check," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-140, Oct 2009. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-140.html

[15] S. Albers and A. Antoniadis, "Race to idle: New algorithms for speed scaling with a sleep state," *ACM Trans. Algorithms*, vol. 10, no. 2, pp. 9:1–9:31, Feb. 2014. [Online]. Available: http://doi.acm.org/10.1145/2556953

[16] C. Imes and H. Hoffmann, "Minimizing energy under performance constraints on embedded platforms: Resource allocation heuristics for homogeneous and single-isa heterogeneous multi-cores," *SIGBED Rev.*, vol. 11, no. 4, p. 49–54, jan 2015. [Online]. Available: https://doi.org/10.1145/2724942.2724950

[17] D. H. K. Kim, C. Imes, and H. Hoffmann, "Racing and Pacing to Idle: Theoretical and Empirical Analysis of Energy Optimization Heuristics," *Proceedings - 3rd IEEE International Conference on Cyber-Physical Systems, Networks, and Applications, CPSNA 2015*, pp. 78–85, 2015.

[18] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang, "Ppep: Online performance, power, and energy prediction framework and dvfs space exploration," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2014, pp. 445–457.

[19] V. Spiliopoulos, S. Kaxiras, and G. Keramidas, "Green governors: A framework for continuously adaptive dvfs," in *2011 International Green Computing Conference and Workshops*, July 2011, pp. 1–8.

[20] K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, June 2012, pp. 213–224.

[21] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan, "Variability in the execution of multimedia applications and implications for architecture," *Proceedings 28th Annual International Symposium on Computer Architecture*, vol. 00, no. June, 2001.

[22] C. J. Hughes, J. Srinivasan, and S. V. Adve, "Saving energy with architectural and frequency adaptations for multimedia applications," in *Proceedings. 34th ACM/IEEE International Symposium on Microarchitecture. MICRO-34*, Dec 2001, pp. 250–261.

[23] J. Suh, C.-T. Huang, and M. Dubois, "Dynamic mips rate stabilization for complex processors," *ACM Trans. Archit. Code Optim.*, vol. 12, no. 1, Apr. 2015. [Online]. Available: https://doi.org/10.1145/2714575

[24] B. Donyanavard, T. Mück, S. Sarma, and N. Dutt, "Sparta: Runtime task allocation for energy efficient heterogeneous manycores," in *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2016, pp. 1–10.

[25] D. De Sensi, M. Torquati, and M. Danelutto, "A reconfiguration algorithm for power-aware parallel applications," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 4, pp. 43:1–43:25, Dec. 2016. [Online]. Available: http://doi.acm.org/10.1145/3004054

[26] J. Li and J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, Feb 2006, pp. 77–87.

[27] S. Baskiyar and R. Abdel-Kader, "Energy aware dag scheduling on heterogeneous systems," *Cluster Computing*, vol. 13, no. 4, pp. 373–383, Dec 2010. [Online]. Available: https://doi.org/10.1007/s10586-009-0119-6

[28] P. Alonso, M. F. Dolz, R. Mayo, and E. S. Quintana-Ortí, "Improving power efficiency of dense linear algebra algorithms on multi-core processors via slack control," in *2011 International Conference on High Performance Computing Simulation*, 2011, pp. 463–470.

[29] N. Kumar and D. P. Vidyarthi, "A green sla constrained scheduling algorithm for parallel/scientific applications in heterogeneous cluster systems," *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 107 – 119, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210537918301124

[30] Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. USA: IEEE Computer Society, 2009, p. 92–99. [Online]. Available: https://doi.org/10.1109/CCGRID.2009.16

[31] J. Kang and S. Ranka, "Dynamic slack allocation algorithms for energy minimization on parallel machines," *J. Parallel Distrib. Comput.*, vol. 70, no. 5, p. 417–430, May 2010. [Online]. Available: https://doi.org/10.1016/j.jpdc.2010.02.005

[32] Q. Zhang, F. Yuan, R. Ye, and Q. Xu, "Approxit: An approximate computing framework for iterative methods," in *Proceedings of the 51st*

Annual Design Automation Conference, ser. DAC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 1–6. [Online]. Available: https://doi.org/10.1145/2593069.2593092

[33] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "Approxann: An approximate computing framework for artificial neural network," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 701–706.

[34] V. Vassiliadis, C. Chalios, K. Parasyris, C. D. Antonopoulos, S. Lalis, N. Bellas, H. Vandierendonck, and D. S. Nikolopoulos, "Exploiting significance of computations for energy-constrained approximate computing," *International Journal of Parallel Programming*, vol. 44, no. 5, pp. 1078–1098, 2016.

[35] H. Hoffmann, "Jouleguard: Energy guarantees for approximate applications," in *Proceedings of the 25th Symposium on Operating Systems Principles*, ser. SOSP '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 198–214. [Online]. Available: https://doi.org/10.1145/2815400.2815403

[36] A. Farrell and H. Hoffmann, "MEANTIME: Achieving both minimal energy and timeliness with approximate computing," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. Denver, CO: USENIX Association, Jun. 2016, pp. 421–435. [Online]. Available: https://www.usenix.org/conference/atc16/technical-sessions/presentation/farrell

[37] S. S. Dayapule, F. Yao, and G. Venkataramani, "Powerstar: Improving power efficiency in heterogenous processors for bursty workloads with approximate computing," in *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019, pp. 175–182.

[38] N. Kulkarni, F. Qi, and C. Delimitrou, "Pliant: Leveraging approximation to improve datacenter resource efficiency," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019, pp. 159–171.

[39] F. Kluge, S. Uhrig, J. Mische, B. Satzger, and T. Ungerer, "Dynamic workload prediction for soft real-time applications," *Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010*, no. Cit, pp. 1841–1848, 2010.

[40] T. K. Hyun-Duk Cho, Kisuk Chung. (2012) Benefits of the big.little architecture. [Online]. Available: http://www.samsung.com/semiconductor/minisite/Exynos/data/benefits.pdf

[41] H. Chung, M. Kang, and H.-D. Cho, "Heterogeneous Multi-Processing Solution of Exynos 5 Octa with ARM big.LITTLE Technology," Tech. Rep., 2013. [Online]. Available: https://www.arm.com/files/pdf/Heterogeneous_Multi_Processing_Solution_of_Exynos_5_Octa_with_ARM_bigLITTLE_Technology.pdf

[42] M. L. Li, R. Sasanka, S. V. Adve, Y. K. Chen, and E. Debes, "The ALPBench benchmark suite for complex multimedia applications," in *Proceedings of the 2005 IEEE International Symposium on Workload Characterization, IISWC-2005*, vol. 2005, 2005, pp. 34–45.

[43] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[44] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, Oct 2009, pp. 44–54.

[45] M. I. Gordon, W. Thies, and S. Amarasinghe, "Exploiting coarse-grained task, data, and pipeline parallelism in stream programs," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XII. New York, NY, USA: Association for Computing Machinery, 2006, p. 151–162. [Online]. Available: https://doi.org/10.1145/1168857.1168877

[46] J. Abella, C. Hernandez, E. Quiñones, F. J. Cazorla, P. R. Conmy, M. Azkarate-askasua, J. Perez, E. Mezzetti, and T. Vardanega, "Wcet analysis methods: Pitfalls and challenges on their trustworthiness," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2015, pp. 1–10.

[47] H. Chung, M. Kang, and H.-D. Cho. (2013) Heterogeneous multi-processing solution of exynos 5 octa with arm big.littletm technology. [Online]. Available: https://www.arm.com/files/pdf/Heterogeneous_Multi_Processing_Solution_of_Exynos_5_Octa_with_ARM_bigLITTLE_Technology.pdf

[48] M. W. Azhar. (2021) Bsc application repository openmp transformation. [Online]. Available: https://github.com/waqarazhar/BAR_OpenMP

[49] BSC. (2021) Bsc application repository. [Online]. Available: https://pm.bsc.es/projects/bar

[50] M. W. Azhar, P. Stenström, and V. Papaefstathiou, "Sloop: Qos-supervised loop execution to reduce energy on heterogeneous architectures," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 4, pp. 1–25, 2017.

[51] M. W. Azhar, M. Pericàs, and P. Stenström, "Sac: Exploiting execution-time slack to save energy in heterogeneous multicore systems," in *Proceedings of the 48th International Conference on Parallel Processing*, ser. ICPP 2019. ACM, 2019. [Online]. Available: https://doi.org/10.1145/3337821.3337865

[52] M. Nejat, M. Manivannan, M. Pericàs, and P. Stenström, "Coordinated management of dvfs and cache partitioning under qos constraints to save energy in multi-core systems," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 246–259, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520302859

[53] M. Nejat, M. Pericas, and P. Stenstrom, "Qos-driven coordinated management of resources to save energy in multi-core systems," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 303–313.

[54] M. W. Azhar, M. Pericàs, and P. Stenström, "Task-rm: A resource manager for energy reduction in task-parallel applications under quality of service constraints," *ACM Trans. Archit. Code Optim.*, vol. 19, no. 1, jan 2022. [Online]. Available: https://doi.org/10.1145/3494537

[55] M. W. Azhar. (2021) Workloads for approx-rm. [Online]. Available: https://github.com/waqarazhar/Approx-RM-Workloads

[56] X. Mei, Q. Wang, and X. Chu, "A survey and measurement study of gpu dvfs on energy conservation," *Digital Communications and Networks*, vol. 3, no. 2, pp. 89–100, 2017.