Hergys Rexha

Energy Aware Runtime Systems for Elastic Stream Processing Platforms

Åbo Akademi University

Energy Aware Runtime Systems for Elastic Stream Processing Platforms

Hergys Rexha

Åbo Akademi University
Faculty of Science and Engineering
Vattenborgsvägen 3, 20500, Åbo

2023

Supervisor

Docent Sébastien Lafond
Faculty of Science and Engineering
Åbo Akademi University
Vattenborgsvägen 3, 20500, Åbo
Finland

Reviewers

Permanent Researcher Laurent Lefèvre
The French Institute for Research in Computer Science (INRIA)
Ecole Normale Supérieure de Lyon
France

Professor Jean-Marc Pierson
Department of Computer Science
University of Toulouse
Toulouse
France

Opponent

Professor Jean-Marc Pierson
Department of Computer Science
University of Toulouse
Toulouse
France

ISBN: 978-952-12-4305-9 (printed)
ISBN: 978-952-12-4306-6 (digital)
Painosalama Oy, Turku, Finland 2023

Abstract

Following an invariant growth in the required computational performance of processors, the multicore revolution started around 20 years ago. This revolution was mainly an answer to power dissipation constraints restricting the increase of clock frequency in single-core processors. The multicore revolution not only brought in the challenge of parallel programming, i.e. being able to develop software exploiting the entire capabilities of many-core architectures, but also the challenge of programming heterogeneous platforms. The question of "on which processing element to map a specific computational unit?", is well known in the embedded community. With the introduction of general-purpose graphics processing units (GPGPUs), digital signal processors (DSPs) along with many-core processors on different system-on-chip platforms, heterogeneous parallel platforms are nowadays widespread over several domains, from consumer devices to media processing platforms for telecom operators. Finding mapping together with a suitable hardware architecture is a process called design-space exploration. This process is very challenging in heterogeneous many-core architectures, which promise to offer benefits in terms of energy efficiency. The main problem is the exponential explosion of space exploration. With the recent trend of increasing levels of heterogeneity in the chip, selecting the parameters to take into account when mapping software to hardware is still an open research topic in the embedded area. For example, the current Linux scheduler has poor performance when mapping tasks to computing elements available in hardware. The only metric considered is CPU workload, which as was shown in recent work does not match true performance demands from the applications. Doing so may produce an incorrect allocation of resources, resulting in a waste of energy. The origin of this research work comes from the observation that these approaches do not provide full support for the dynamic behavior of stream processing applications, especially if these behaviors are established only at runtime. This research will contribute to the general goal of developing energy-efficient solutions to design streaming applications on heterogeneous and parallel hardware platforms. Streaming applications are nowadays widely spread in the software domain. Their distinctive charac-

teristic is the retrieving of multiple streams of data and the need to process them in real time. The proposed work will develop new approaches to address the challenging problem of efficient runtime coordination of dynamic applications, focusing on energy and performance management.

Sammandrag

Efter en oföränderlig tillväxt i prestandakrav hos processorer, började den flerkärniga processor-revolutionen för ungefär 20 år sedan. Denna revolution skedde till största del som en lösning till begränsningar i energieffekten allt eftersom klockfrekvensen kontinuerligt höjdes i en-kärniga processorer. Den flerkärniga processor-revolutionen medförde inte enbart utmaningen gällande parallellprogrammering, m.a.o. förmågan att utveckla mjukvara som använder sig av alla delelement i de flerkärniga processorerna, men också utmaningen med programmering av heterogena plattformar. Frågeställningen ”på vilken processorelement skall en viss beräkning utföras?” är väl känt inom ramen för inbyggda datorsystem. Efter introduktionen av grafikprocessorer för allmänna beräkningar (GPGPU), signalprocesserings-processorer (DSP) samt flerkärniga processorer på olika system-on-chip plattformar, är heterogena parallella plattformar idag omfattande inom många domäner, från konsumtionsartiklar till mediaprocesserings plattformar för telekommunikationsoperatörer. Processen att placera beräkningarna på en passande hårdvaruplattform kallas för utforskning av en designrymd (design-space exploration). Denna process är mycket utmanande för heterogena flerkärniga arkitekturer, och kan medföra fördelar när det gäller energieffektivitet. Det största problemet är att de olika valmöjligheterna i designrymden kan växa exponentiellt. Enligt den nuvarande trenden som förespår ökad heterogeniska aspekter i processorerna är utmaningen att hitta den mest passande placeringen av beräkningarna på hårdvaran ännu en forskningsfråga inom ramen för inbyggda datorsystem. Till exempel, den nuvarande schemaläggaren i Linux operativsystemet är inkapabel att hitta en effektiv placering av beräkningarna på den underliggande hårdvaran. Det enda mätsättet som används är processorns belastning vilket, som visats i tidigare forskning, inte motsvarar den verkliga prestandan i applikationen. Användning av detta mätsätt vid resursallokering resulterar i slöseri med energi. Denna forskning härstammar från observationerna att dessa tillvägagångssätt inte stöder det dynamiska beteendet hos ström-processeringsapplikationer (stream processing applications), speciellt om beteendena bara etableras vid körtid. Denna

forskning kontribuerar till det allmänna målet att utveckla energieffektiva lösningar för ström-applikationer (streaming applications) på heterogena flerkärniga hårdvaruplattformar. Ström-applikationer är numera mycket vanliga i mjukvarudomän. Deras distinkta karaktär är inläsning av flertalet dataströmmar, och behov av att processera dem i realtid. Arbetet i denna forskning understöder utvecklingen av nya sätt för att lösa det utmanade problemet att effektivt koordinera dynamiska applikationer i realtid och fokus på energi- och prestandahantering.

Acknowledgements

Well, this has been a long journey that led to the creation of this work, and it was influenced by many people and factors until it was finished.

First a big gratitude, goes to my supervisor Sébastien Lafond, for his continuous support, patience, and belief during these years. During this time I had the opportunity to work with him and learn a lot professionally and at the human level. Also, I would like to thank Professor Johan Lilius for the nice discussions and the enjoyable meetings of the Stream Computing Group, which I remember with great nostalgia.

I would like to thank Professor Jean-Marc Pierson and Permanent Researcher Laurent Lefèvre for reviewing my thesis and giving great suggestions for improving the quality of the final material. Additional thanks go to Professor Jean-Marc Pierson for accepting to act as my opponent in the doctoral defense. This research was published in a collection of papers over the years, which has the contributions of several people, to whom I am grateful for their assistance. Among colleagues, I would like to give a special thanks to Simon Holmbacka whose contribution as co-author and advisor has been of great help. I also want to express my gratitude to other colleagues from the Embedded System Laboratory like, Sudeep Kanur, Wictor Lund, Georgios Georgakarakos, Srboljub Stepanovic, Tanwir Ahmad, the discussion with them has had a great impact in the development of my work.

I would like to acknowledge the wonderful support received during these years from the members of the Faculty of Natural Sciences and Engineering, the administrative and technical part. Among others, I would like to thank Christel Engblom, Minna Carla, Karl Rönnholm, Marat Vagapov, Pia-Maria Kallio for always patiently helping in solving issues.

I am really grateful for the scholarship received from the Erasmus Mundus EUROWEB+ program which in the first part made it possible to start this research and later for the doctoral funding from Åbo Akademi.

Finally, I would like to express my warmest thanks to my parents and brother for their support and encouragement. Most of all, my infinite thanks to my wife Esmeralda for her patience, love, and her confidence in me.

Åbo, July 2023
Hergys Rexha

List of Original Publications

1. Hergys Rexha, Simon Holmbacka, and Sébastien Lafond. Core level utilization for achieving energy efficiency in heterogeneous systems. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 401–407, 2017.

2. Hergys Rexha and Sébastien Lafond. Exploring energy efficiency model generalization on multicore embedded platforms. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 494–498, 2018.

3. Hergys Rexha, Sébastien Lafond, and Karol Desnos. Energy-efficient actor execution for sdf application on heterogeneous architectures. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 486–493, 2018.

4. Hergys Rexha and Sébastien Lafond. Energy efficiency platform characterization for heterogeneous multicore architectures. In Annika Wolff, editor, *Proceedings of the 6th International Conference on ICT for Sustainability, ICT4S 2019, Lappeenranta, Finland, June 10-14, 2019*, volume 2382 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

5. Hergys Rexha, Sébastien Lafond, Giovanni Rigazzi, and Jani-Pekka Kainulainen. Towards very low-power mobile terminals through optimized computational offloading. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2020.

6. Hergys Rexha and Sébastien Lafond. Data collection and utilization framework for edge ai applications. In *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, pages 105–108, 2021.

List of Other Publications

1. Ivan Porres, Tanwir Ahmad, Hergys Rexha, Sébastien Lafond, and Dragos Truscan. Automatic exploratory performance testing using a discriminator neural network. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 105–113, 2020.

2. Ivan Porres, Hergys Rexha, and Sébastien Lafond. Online gans for automatic performance testing. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 95–100, 2021.

3. Sebastian Robitzsch, Josep Ribes, Andre S. Gomes, Hergys Rexha, Luis Cordeiro, Mohamad Kenan Al-Hares, Marius Corici, and David Gomez-Barquero. Under trial: Evolved service-based architecture platform for mobile telecommunication networks. In *2022 IEEE Future Networks World Forum (FNWF) - Main Track 3: 5G and Future Networks' Trials, Experimental Results and Deployment Scenarios*, 2022.

Contents

Part I

RESEARCH SUMMARY

Chapter 1

Introduction

Digital computers have been the first to implement every technological and scientific advancement from the 80' to modern days. Their speed has been increasing ever since at a rapid pace as predicted by the Moore Law in 1970. In reality, Gordon Moore predicted the transistor count inside chips will proceed to grow exponentially, defining still today's industry trends, not the operational speed!

With the increase of transistor count inside the chip, the operating frequency also increased giving way to various interpretations of Moore's law that might predict the speed of computers. The increase in chip frequency produced a natural increase in heat generation, which was kept under control with the lowering of the supply voltage. The supply (currently it has reached at 1.29V) voltage reduction has its own limits for transistors to be operating correctly, so the necessity to manage the heat resulted in stagnation of the operating frequency at 2-5 GHz in the mid-2000s. So the only way to meet the demands for more computational power was by squeezing more computational logic, inside computing cores. The first dual-core chip was introduced by IBM in 2001 (Power 4), and later many variations of multicore chips came by, like the 64-core homogeneous chip TILE64, or *heterogeneous* versions like the Cell BE used in PS3.

The multi-core revolution not only brought in the challenge of parallel programming, i.e. being able to develop software exploiting the entire capabilities of many-core architectures, but also the challenge of programming heterogeneous platforms. The question of *"on which processing element to map a specific computational unit?"*, is well known in the embedded community. With the introduction of general-purpose graphics processing units (GPGPUs), digital signal processors (DSPs) along with many-core processors on different system-on-chip platforms, and heterogeneous parallel platforms are nowadays widespread over several domains, from consumer devices to media processing platforms for telecom operators. Finding mapping to-

gether with a suitable hardware architecture is a process called design-space exploration. This process is very challenging in heterogeneous many-core architectures, which promise to offer benefits in terms of energy efficiency. The main problem is the exponential explosion of the space exploration. Several examples from the industry [1] show that there is a trend of increasing levels of heterogeneity in the chip. Selecting the parameters to take into account when mapping software to hardware is still an open research topic in the embedded area.

1.1 State of the art

For example, the current Linux scheduler has poor performance when mapping tasks to computing elements available in hardware. The only metric considered is CPU workload, which as it was shown in recent work [1] does not match true performance demands from the applications. Doing so may produce an incorrect allocation of resources, resulting in a waste of energy.

Because there is an actual lack of efficiency in the current scheduling decisions in the context of heterogeneous systems as shown in Figure 1.1, the focus of this research is to design a runtime system that could efficiently allocate resources to tasks according to their need, focusing on energy efficiency.

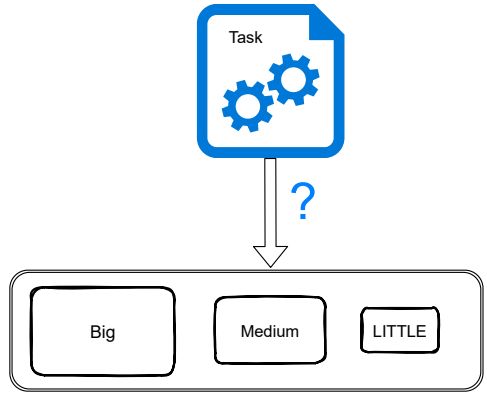

Figure 1.1: Allocating hardware resources to tasks, not energy efficient in current schedulers

At the same time in many cases, signal processing systems and streaming applications can be described at several levels of abstraction using the dataflow programming paradigm. For example, the concept of Synchronous Dataflow (SDF) graphs for streaming applications was developed and used extensively by Lee and Messerschmitt [2] as it is a modeling concept suited to describe parallelism. Computations are executed in actors and data is exchanged through FIFO buffers. For most of the current streaming applications, dynamic behavior has become very common and needs to be efficiently supported by the programming paradigm. For example, video coding and recent telecommunication [3] standards use adaptive algorithms

[1] http://www.engadget.com/2017/02/27/mediatek-helio-x30-deca-core-processor/

that cannot be fully described and executed unless dynamic behavior is supported. Since the dataflow model offers a clear separation of computations from data transfers, it is a natural choice as a programming paradigm in our research.

1.2 Research Question

Several academic and commercial frameworks supporting the modeling, design, and analysis of streaming applications were proposed over the last decade. The prevalent academic tool is the Ptolemy [4], a framework developed at the University of California at Berkley. It supports the modeling and design of heterogeneous systems using a variety of dataflow-based models of computation. In the industry, the LabView software commercialized by National Instruments is probably the most popular one. It uses the dataflow programming language G to model parallel applications. Several other frameworks exist, including Simulink, CoFluent Design, StreamIt [5], PEACE [6], Graphiti, and PREESM [7].

The origin of this research work comes from the following question: how to better support the dynamic behavior of stream processing applications, especially if these behaviors are established only at runtime?

The community in the Embedded Systems Laboratory at Åbo Akademi University has worked for the past years with dataflow-based languages, runtime systems, and many-core platforms, focusing recently on heterogeneous architectures. Previous work in this community developed the notion of crossover point [8] which can be determined with a combination of static and runtime analysis and is based on the chunk size of work used in the kernel. This determines when to move the mapping of a specific kernel from the CPU to the GPU. Previous activities also focused on the development of accurate power models [9, 10] and new approaches for runtime energy management [11, 12, 13] of parallel applications deployed on heterogeneous platforms.

The concept of *Fitness* of computation on a particular processing unit has been largely studied. A recent work [1] shows that the type of instructions to be executed is a key factor in determining the execution strategy on big.LITTLE platforms. In [14] it is shown through experimental work that the load level of a task is not the right metric for determining an energy-efficient strategy for heterogeneous platforms. Several factors should be considered for making the optimal energy-efficient decision when executing on heterogeneous many-core platforms.

This research work is a direct and logical continuation of the previous research works, presented above, conducted at Åbo Akademi University in

cooperation with researchers from several Finnish and foreign universities and the industry, pushing the research effort into the development of runtime supports for elastic stream processing applications.

1.3 Scientific objectives and expected impact

1.3.1 Scientific objectives

The broad goal of the research is to design a method for achieving portable and efficient streaming applications. To achieve this goal, the following question needs to be continuously answered at runtime: *where to map a specific workload on a given architecture with the awareness that workload execution should reach the intended requirements of performance, energy efficiency and/or reliability?*

Hypothesis: The main hypothesis is based on the fact that techniques and methods developed for the embedded world could be extended to the streaming domain. So, tools previously used in the design process for embedded systems could be developed further to support the dynamic behavior of stream computing systems.

Expected objectives: Overall, this research will contribute to developing the adoption of dataflow-oriented programming languages. More concretely the objectives are the following:

- Provide runtime energy management for dynamic streaming applications.
- Provide runtime performance management for dynamic streaming applications.
- Provide runtime support for flexible composition and concurrent execution of several dynamic streaming applications.

Expected research results: This work is expected to develop methods and techniques to efficiently manage at runtime the dynamic behavior of stream processing applications. It will participate in demonstrating the large impact dataflow-oriented programming languages can have on the design flow of stream processing applications.

1.3.2 Effects and impact beyond academia

Having a widely adopted stream programming paradigm and runtime framework would affect most consumer electronics industries (media processing devices, mobile phones, etc.) and telecommunication industries (base stations, media gateways, etc.). The work will participate in the middle-term

objectives of making dataflow-based application design a) optimal for programming parallel, heterogeneous processors, b) portable, and c) efficient and scalable.

1.4 Research Contributions

This research will contribute to the general goal of developing energy-efficient solutions to design streaming applications on heterogeneous and parallel hardware platforms. Streaming applications are nowadays widely spread in the software domain. Their distinctive characteristic is the retrieving of multiple streams of data and the need to process them in real time. Examples of such applications can be found in application domains such as banking systems, video conferencing, multimedia processing, Telecom(5G), etc.

The main motivation for this research arose from the observation that current solutions do not provide full support for the dynamic behavior of stream processing applications. Therefore the proposed work will develop new approaches to address the challenging problem of efficient runtime coordination of dynamic applications, focusing on energy and performance management. As ICT infrastructure in Europe in 2012 consumed 1,6% of the world's electrical energy consumption [15], this work could lead to an important impact in terms of worldwide energy consumption.

1.5 Thesis Organization

The thesis is organized into six chapters. Chapter I is an introduction to the research area with an analysis of the state-of-the-art work, research questions posed in the thesis, and research contributions of the thesis. Chapter II contains a description of the main notions used throughout the research in order to develop energy efficiency models. In Chapter III there is a description of the mechanism used in the research for controlling the load during task execution. Chapter IV, is the heart of the research, with original contributions through the research in order to answer research questions set at the beginning. Chapter V is a brief summary of the original publications and a summary of each paper. Chapter VI ends with a discussion of the conclusions and future work which develop from the thesis work.

Chapter 2

Background

2.1 Definition of Energy and Power in computing platforms

Through the process of evolution of electronic and computer systems, the progress was defined in terms of density and speed. Increased density would mean, different geometries of semiconductor devices. Processing speed was also affected by this increase in density, more computation resources, and more data storage in chips, leading to higher levels of parallelism and computations completed per unit of time. Managing the power dissipation of computing systems has been a challenge of architects for a long time. This is a problem found in many domains of computing: from large-scale data centers [16] [17], to high-end servers [18], going through battery-operated devices such as smartphones, laptops, and even IoT devices [19, 20]. To maintain the trend of higher performance as the number of transistors per chip rises, designers have resorted to more elaborate processor designs (pipelining, superscalar, SMT) and to high clock frequencies. Unfortunately, power requirements have grown exponentially as chip density and clock frequency have risen. One way to control power density is to use more of the chip area for cache memory. Memory transistors are smaller and have a power density an order of magnitude lower than that of logic. As chip transistor density has increased, the percentage of chip area devoted to memory has grown and is now often half the chip area. Even so, there is still a considerable amount of chip area devoted to the processing logic. Energy is often considered one of the most fundamental metrics, especially in battery-operated devices, where the availability of it is limited to the capacity of the battery. But even in non-mobile systems energy consumption is ranked as one of the leading operating costs which means there is a high need for the reduction of it. Power which is measured in Watts(W) is the rate of energy dissipation and is the metric that relates to the current delivery and voltage regulation

inside the chip. Another metric of interest is power density which is the dissipated power per unit area and is studied for thermal analysis; 200W dissipated in $4cm^2$ surface could be very difficult to cool down.

2.1.1 Power efficiency compared to Energy efficiency

In the literature, you can find in usage the terms of *Power efficiency* and *Energy efficiency* as the same concept. However there is a clear distinction between the previously mentioned terms. Power in a CMOS circuit is defined by the voltage the circuit operates and the current flowing as a function of time.

$$P(t) = V(t)xI(t) \tag{2.1}$$

where V and I are instantaneous values of voltage and current. Generally, in many systems, the voltage is constant so the power depends on the current drawn by the system, such that the power follows the change of the current. The energy consumed by the system over a period from t=0 to t=T would be:

$$E = \int_t^T P(t)\ dt \tag{2.2}$$

and is denoted by the area of the graph below the power curve P(t) like in Fig 2.1. Two systems having different power dissipation P(t) and P'(t) consume different amounts of energy, respectively E and E'. From the Figure 2.1, we can tell that system S is more energy efficient and power efficient than system S$'$. The discussion is different in the comparison between system S and S$''$ in Fig 2.2. The power characteristic of system S$''$ shows that the power values are under those of system S but the time S$''$ takes to complete the work is larger and the energy is calculated over the interval 0 to T$''$ as:

$$E = \int_t^{T''} P''(t)\ dt \tag{2.3}$$

Clearly, we can state that if E$'' <$ E, then S$''$ is more energy efficient than S because the amount of work is not connected to completion time. The discussion for power efficiency depends on metrics like peak power and average power. If we account for the whole interval of time from 0 to T$''$ then the average power is:

$$Pavg(S) = \frac{E}{T''} \tag{2.4}$$

$$Pavg(S'') = \frac{E''}{T''} \tag{2.5}$$

The peak power is the highest value the power curve should not exceed while the system is running. It's actually imposed by the system design. If

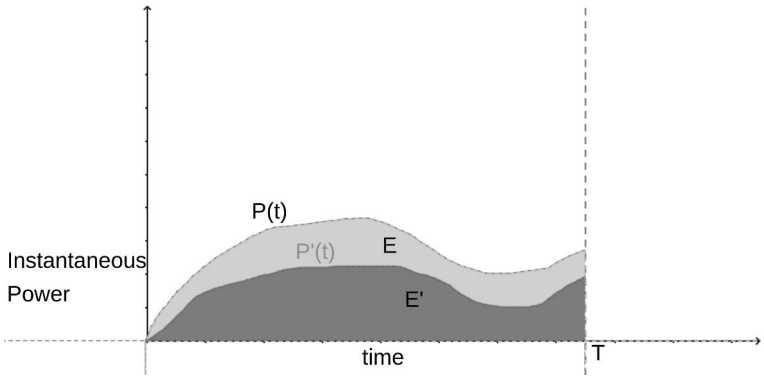

Figure 2.1: Relation between power and energy for two power curves

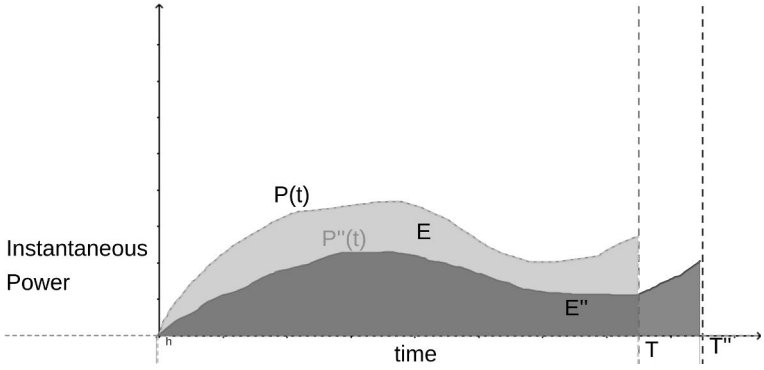

Figure 2.2: Power and energy efficiency for systems with different competition times.

this metric is considered system S″ power values are under system S, so is more power efficient. Peak power constraints can be set for 2 reasons:

- constraints in the actual power that the supply can provide
- maximum temperature that the system can support

2.1.2 Power-performance tradeoffs

Figure 2.2 shows the trade-off that exists between performance and energy which has many practical implications. Especially in the domain of mobile devices, where most of the time the focus is not only on pure computational performance but also on network connectivity, managing a high variety of workloads, and battery life. So the push to deliver more and more performance has shed new and strong light on energy and power. Many metrics are proposed in the literature for grasping the inverse relationship between

power and performance. The *energy-delay product*(EDP) [21] is used often in order to evaluate systems taking into account both energy and performance, and not discriminating systems only by the power dissipation. For the systems analyzed in Equation 2.5

$$EDP(S) = E \times T$$

$$EDP(S'') = E'' \times T''$$

A second metric of evaluation is *performance-per-watt* used to characterize the energy efficiency [22]. In this metric, we can notice sometimes actually the calculation of 1/E. The performance can be interpreted in different ways, such as; throughput, latency, or frequency of computations. If the performance is accounted as the inverse of latency then we would calculate:

$$performance - per - watt = \frac{1}{T} \times \frac{1}{P} = \frac{1}{E}$$

A third metric is *energy-delay squared*(ED^2P) where the energy consumed by the system is multiplicated with the square of the delay, giving this way more emphasis to the performance.

2.2 CMOS circuits power

CMOS circuit power is divided into several categories: dynamic power, leakage power (or static power), and short circuit power. We will focus next on the two major sources of power dissipation: dynamic and leakage.

2.2.1 Dynamic Power

The dominant part of power in CMOS systems is dynamic power, which is related to the switching activity of the transistors and the subsequent charging of the load capacitance. The dynamic power formula is as below:

$$P = ACV^2F$$

C is the aggregate load capacitance of the circuit, while A and F are respectively the switching activity factor of a node (probability that the node will change the state), and the operating frequency. The C parameter can be reduced by using small transistors and small interconnecting wires in non-critical parts of the circuit. By using small cores instead of large processors, or dedicated caches instead of large ones we can impact lowering the C factor.

Voltage V is the supply voltage of the transistors which for decades has been decreasing steadily as a result of advancements in process technology.

On average for each technology generation the operational voltage has decreased by 30% [23]. Based on graphs in [24] we can see that the supply voltage V_{DD} has been reduced to 0.65V for the 5nm process technology with the prospective of 0.6V supply voltage for the 2nm node. According to the ITRS projection [25] there will be a gradual decrease of both the working voltage and threshold voltage of the transistor as shown in Figure 2.3.

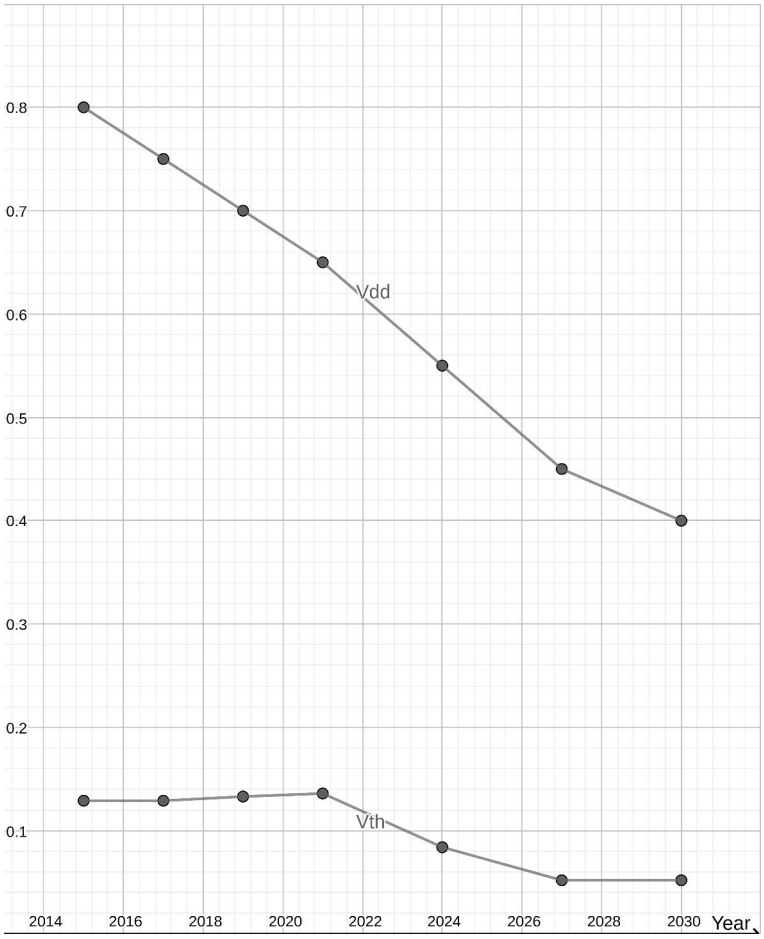

Figure 2.3: ITRS specification for supply voltage Vdd and the threshold voltage of the transistor

2.2.2 Static Power

The overall power dissipation in CMOS circuits is given by the following formula.

$$P = ACV^2 f + V I_{\text{leak}}$$

The first part of the equation represents the dynamic power of the circuit and is spent by charging and discharging the capacitive load of the transistors inside the CPU. The second term shows the other part of power, which is called leakage power or static power. This part of power dissipation is lost due to the presence of the leakage current (I_{leak}) in the CMOS transistor. While for dynamic power by reducing the working voltage we could have benefits in terms of power saved, and hide the reduction of speed with techniques such as parallelism and pipeline, in the static case this technique won't provide a reduction in the leakage power. The leakage current is a combination of subthreshold current and gate-oxide leakage.

$$I_{leak} = I_{sub} + I_{ox}$$

In [26] authors show how the subthreshold leakage depends on the threshold voltage and supply voltage.

$$I_{\text{sub}} = K_1 W e^{-V_{th}/nV_\theta}(1 - e^{-VN_\theta})$$

where K_1 and n are derived experimentally, W is the gate width, and V_θ is the thermal voltage. In normal room temperature V_θ is about 25.85mV and increases linearly with the temperature. If the subthreshold current increases in the chip, then the temperature of the device will also increase which will provide a further increase in the leakage. According to the above equation, there are two ways to decrease I_{sub}. One way would be to decrease the source voltage possibly setting it to zero so the exponential in parenthesis will go to one and the entire factor goes to zero. The second way would be to increase the threshold voltage V_{th}, which will provide a significant decrease in the I_{sub} current. But the relation between f and V_{th} tells that every increase of the voltage will decrease the operating frequency, so lower the speed.

$$f \propto (V - V_{\text{th}})^\alpha / V$$

Gate width W is another parameter that influences the subthreshold current, and in the calculations, the W represents the combined widths of the transistors in a chip. For the gate-oxide current according to [26] the key influencing factors would be:

$$I_{\text{ox}} = K_2 W \left(\frac{V}{T_{\text{ox}}}\right)^2 e^{-\alpha T_{\text{ox}}/V}$$

where K_2 and α are experimentally derived. T_{ox} is the oxide thickness in the transistor and will scale down proportionally with the feature process. Taking into consideration that both subthreshold and gate oxide depend on the number of transistors, techniques that maintain the performance without increasing the density in the chip would be a solution in low-power architectures.

2.2.3 Reducing the leakage power

Scaling transistor size with the newer generation of process technology has been followed by a continuous reduction of the source voltage V, with the aim of reducing the dynamic power dissipation. Following that reduction, also the V_{th} scaled down bringing an increase in the static power dissipation. Currently, the static part of power is becoming more and more significant, which brings attention to the right management of this part of power dissipation. The right addressing of this issue is particularly of paramount in handheld devices which are supposed to be "ON", but not active all the time. In this context minimizing the static power is more important than focusing only on dynamic energy consumption. In practice, there are some techniques used during chip manufacturing such as the use of multiple supply voltages [27], multiple threshold voltages [28], transistor stacking [29] and power gating [30].

More significant power reductions are being archived with new transistor design structures like gate-all-around (GAAFET) transistors [31]. GAAFET's advantages over current FinFET transistors includes lowered leakage current (as gates are present on all four sides of the channel), as well as the ability to adjust channel width for higher performance or lower power consumption.

As discussed in [32] the GAA-FinFET exhibited superior SCE(Short Channel Effect) characteristics thanks to the improved sub-channel leakage suppression by the narrow fin. The GAA-FinFET (compared to conventional FinFET) can be employed as a knob for suppressing SCE as well as for improving series resistance. Furthermore, while the device structure of GAA-FinFET is very similar to that of conventional FinFET, there are several flavors that will be considered in production, like GAA vertically stacked nano sheets(NS) FETs, and Forksheet(FS) FETs, culminating to N/PMOS devices stacked on top of each other to form complementary FETs(CFETs) [33]. Which is being consodered as the ultimate CMOS architecture [33].

Delivering high current(data-center SoCs might draw between 100 to 200 amperes) to a huge number of transistors is becoming one of the major bottlenecks in high-performance SoC design. As the size of transistors decreases continuously, the interconnects that supply them with current must be packed ever closer and be made ever finer, on the other hand this increases the resistance and raises the power dissipation. This matter clearly impacts negatively the efficiency of the power delivery system. Without a big change in the way electrons get to and from devices on a chip, it won't matter how much smaller we can make transistors.

As a partial solution to this problem chip manufacturers [31], are considering major advancements in the power delivery of the new transistor

design, called **back-side power delivery**.

The back-side PDN(Power Delivery Network) has the additional advantage of being physically separated from the signal network, so the two networks no longer compete for the same metal-layer resources. It also means that the metal layer characteristics no longer need to be a compromise between what power routes prefer (thick and wide for low resistance) and what signal routes prefer (thin and narrow channels). You can simultaneously tune the back-side metal layers for power routing and the front-side metal layers for signal routing and get the best of both worlds.

This, in turn, will enhance transistor performance and reduce their power consumption. Also, back-side power deliver eliminates some potential interference between data and power connections.

2.2.4 Thermal influence in MPSoCs power dissipation

From the above descriptions of the I_{leak} formula, we see that there is a cyclic relationship between power and temperature. The thermal situation of the chip is dependent on the power dissipated and the stacking of the transistors. On the other hand, if the temperature is high, then the static power dissipation will increase, bringing more heat to the chip. To assess the interaction between temperature and power dissipated during workload execution, we run experiments on two popular MPSoCs from INTEL and ARM. The workloads were chosen as representative of the mobile domain from a popular suite like EEMBC. For CPU testing is used CoreMarkPro[1] set of workloads, which is composed of an integer and floating-point workloads, meant to exercise different components of the CPU in multiple threads.

We use 1000 iterations of the equation solver workload for exercising the MPSoCs with only the active fan as a cooling system. We monitor the current consumed during the execution with an oscilloscope. The results of the current measurement for the two cases are shown in Figures 2.5 and 2.4. The red frame shows the interval during which the workloads are executed. For the ARM case (Figure 2.4), before the workload starts we see a current consumption of 0.6A. After the start of the workload, the current consumption reaches levels of 2.2A, and continuously increases up to 3.2A, a value at which the temperature of the chip is so high that could bring physical damage to the semiconductor. As a protective measure, the board is switched off as can be noticed at the end of the measuring interval. As the same workload is executed during the window interval the continuous increase of power dissipation is attributed to the raise of chip temperature which produces additional static power for the level of approximately 5W.

[1] http://www.eembc.org/coremark/index.php?b=pro.htm

The same experiment is repeated with the INTEL MPSoC (Figure 2.5), but in this case, the current remains constant during the workload execution showing better thermal management.

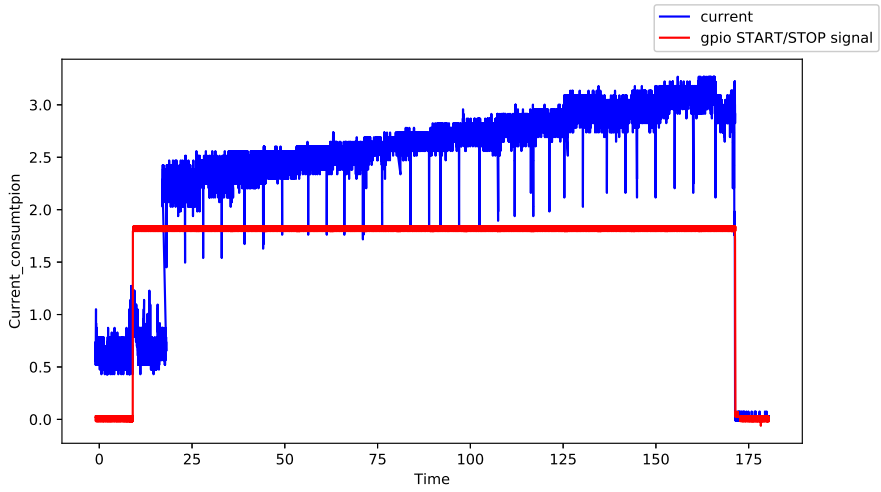

Figure 2.4: Measurement of current consumption in the ARM chip during the execution of the workload

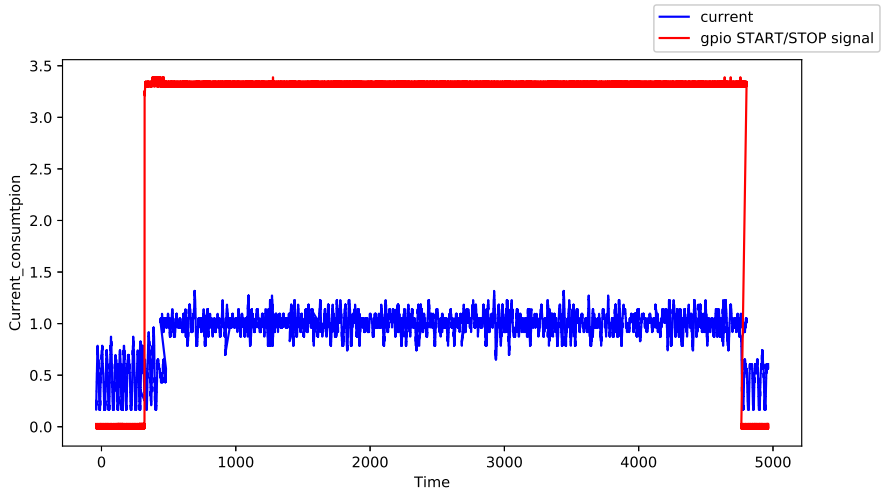

Figure 2.5: Measurement of current consumption in the INTEL chip during the execution of the workload

For having an explanation of the increase of static power we measure the temperature increment on each chip for equation solver workload, while the experiments are conducted in a highly refrigerated environment as explained

in [34]. From Figure 2.6 we can notice the continuous increase of the temperature while the workload is in execution. While in the case of INTEL (Figure 2.7) the temperature reaches a stable value while not increasing anymore. One explanation for the change in the static power component could be the presence of the double of cores in the ARM chip.

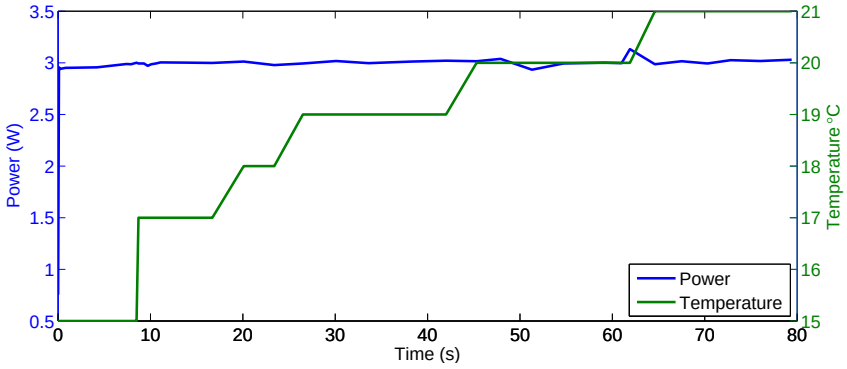

Figure 2.6: Measurements of the linear workload execution on ARM chip

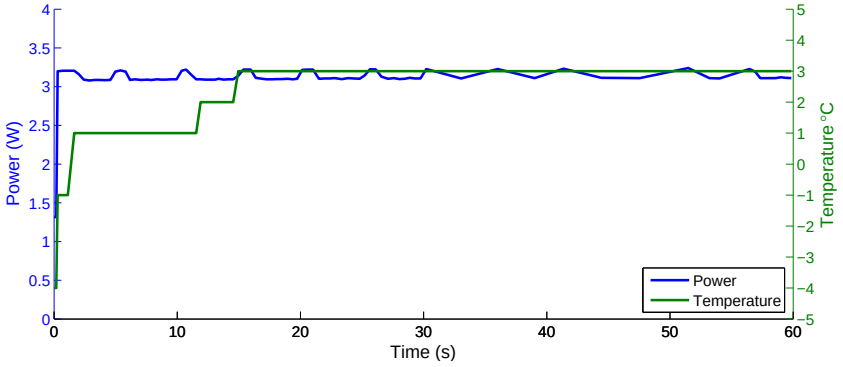

Figure 2.7: Measurements of the linear workload execution on INTEL chip

2.2.5 Energy consumption in multi-core chips

In 2004, the main CPU producers abandoned the continuous increase of the clock frequency, as a result of the power density reached inside a chip. Energy can be saved by using a design where work can be distributed among slower, low-voltage CPU cores that can work together in a parallelized task. The main reason for not working with low voltage would be the increase in leakage energy. The amount of energy needed to complete a task is divided

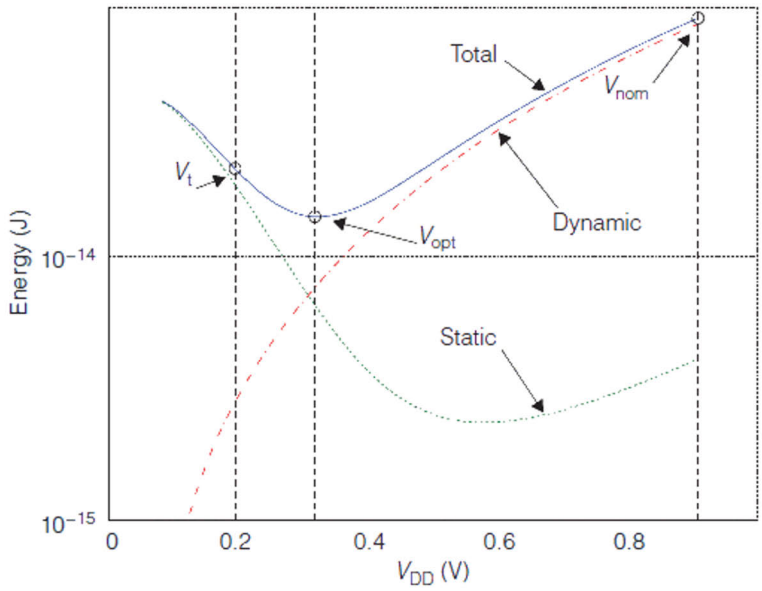

Figure 2.8: Total, static, and dynamic energy across Vdds [35]

into two categories, *dynamic* and *static*.

$$E_{total} = E_{dynamic} + E_{static} = (CV_{DD}^2 + I_{leak}V_{DD})T_{task}$$

The dynamic energy is linked with the task being executed, while the static energy comes from the presence of leakage current during the time the task is executed. The static part of the energy consumed during workload execution is the main limit to the level of energy efficiency that can be achieved. When the source voltage (V_{DD}), is bigger than V_T the frequency scales linearly with the source voltage. As V_{DD} approaches V_T the frequency scales exponentially since the transistor is no anymore in the fully-activated zone. As the operating voltage is lowered the presence of static energy increases because the task requires more time to complete. As it can be seen in Figure 2.8, at a certain level of the operating voltage the static energy is larger than the dynamic energy. According to [35] the operating voltage for which the energy is minimal is called V_{opt}, found by having the derivatives of static and dynamic energy with respect to V_{DD}.

2.3 The notion of heterogeneity

In the past two decades as continuous power constraints were placed in front of chip designers, the focus was set on design styles that could guarantee better power solutions. The multi-core design secured certain power/performance opportunities which satisfied some use cases. Another area of

architectural design could be explored, by using processing elements that have different characteristics of power/performance ratio inside a single chip. These techniques include:

1. heterogeneous on-chip parallelism in which processing elements have the same ISA, but with different configurations or microarchitectures.

2. heterogeneous on-chip parallelism with different processing elements with different ISAs.

3. heterogeneous parallelism with hardware accelerators for special computations.

2.3.1 Microarchitectural Heterogeneity

The simplest form of heterogeneous parallelism is incorporating inside a single chip, cores that have different configurations or microarchitectural designs. Some cores could have the same basic design but work at different frequencies for having different levels of efficiency. One of the first examples of single-ISA heterogeneity is ARM's big.LITTLE approach [36]. This design includes a "big" power-hungry core type for achieving high performance (e.g. ARM Cortex-A15) and "LITTLE" energy efficient low power core (e.g. ARM Cortex-A7), organized each into groups of cores called *cluster*. Both core types have the same architecture but differ in the microarchitecture implementation which gives them a different power and performance behavior. Cache memory and communication are implemented in such a manner that allows fast switching of tasks between cores, and clusters are organized in such a way that all cores can be used at the same time. More recent examples [37] try to increase the number of clusters present inside a chip to provide the better granularity of the power and performance selection. In this case, the first gear of the CPU subsystem is a high-performance (HP) ARM Cortex-A78 at 3GHz that is designed with high-speed library cells. The second gear is made of the ARM Cortex-A78 clocked at 2.6GHz, and designed for balanced performance(BP). The third gear of the subsystem is composed of four ARM Cortex-55 which are designed to provide high efficiency (HE). The promised achieved power and performance behavior is shown in Figure 2.9

2.3.2 Architectural Heterogeneity

While a single-ISA chip is a natural way of having heterogeneity and achieving better energy efficiency through different configurations or designs of cores inside a chip, a more aggressive way to exploit heterogeneity is by having multiple-ISA computational units. While using cores with different architectures is a natural way for architectural heterogeneity, the workload

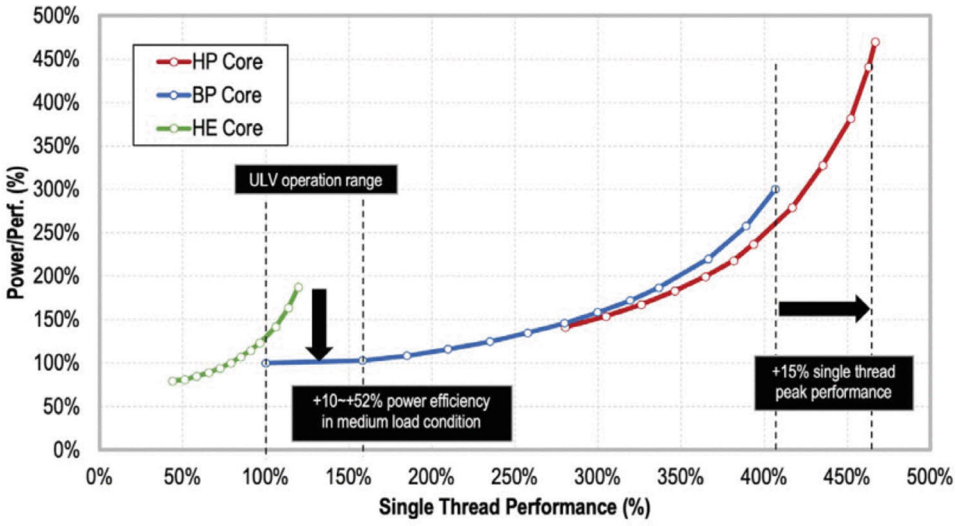

Figure 2.9: Tri-cluster implementation power/efficiency [37]

combination of CPU and graphical processing unit(GPU), has reached good levels of adoption. GPUs were first introduced for workloads that have high levels of parallelism and regularity, such as gaming, and graphics. While at first the power dissipation of such units could seem very high (e.g. 200W) and not possible to reach energy efficiency, the case is that by providing parallel single instruction multiple threads (SIMT) the calculation overhead is reduced for phases like fetch and decode. According to [38] the GPUs are an opportunity to further increase the performance while still staying within the power limits, providing a potential 10x reduction to the energy-per-operation.

2.4 Hardware techniques for energy management

As showed before the dynamic power is related to parameters such as C, V, and f, increasing the preasure on the working voltage in order to leaverage the maximum power reduction by adjusting the voltage (V). Two techniques are on the forefront of the power management with hardware techniques:

- Dynamic Voltage and Frequency Scaling (DVFS)
- Dynamic Power Management (DPM)

2.4.1 Dynamic Voltage and Frequency Scaling (DVFS)

In computing systems, applications do not require the highest performance of the CPU continuously. Often the task CPU is running completes before its deadline, and for the time remaining (called *slack*), the CPU enters low power mode. The intuition behind DVFS is that by reducing the speed of the CPU during execution, the task could finish with zero slack, and the source voltage of the CPU can be reduced to save power. By lowering at the same time frequency (f) and voltage (V) the dynamic power would have a cubic reduction, while the energy consumed will have quadratic reduction.

DVFS in multiple processors system on chip(MPSoC)

With the advent of multi-core era, it is important to understand how to apply dynamic voltage scaling to be even more influential in reducing power dissipation inside the chip. It is a design decision with three alternatives: apply DVFS to the chip level, use voltage scaling separately for each core, or set group of cores which share the same clock domain. Using core level DVFS would mean having multiple clock domains and a synchronizer circuit between them. The positive side of having multiple clock domains would be to break the long clock tree distribution that in high density chip becomes a problem [39]. The other alternative for managing DVFS in chip-multiprocessor(CMP) would be to organize cores into different *voltage/frequency islands*, and cores inside an island to share the same clock domain. As shown by [40] this approach outperforms the others, since the additional complexity of having per-core DVFS does not bring significant benefits in terms of energy savings.

2.4.2 Dynamic Power Management

Even though much attention is focused in techniques for the management of the dynamic power being consumed during workloads execution, with the recent advances of process technology, static power is rapidly increasing [41]. In this context reducing the leakage power of the chip becomes imperative in the pursuit of reducing the overall power dissipation. Techniques used for lowering the dynamic energy consumption, such as frequency and voltage reduction, help in lowering the static power dissipation but still their effect its marginal in the leakage of the transistors. Therefore techniques that focus in reducing the leakage current of the transistors are needed. These techniques belong to the so called DPM (Dynamic Power Management) strategy. The main idea is that the leakage current is proportional to the number of transistors that are powered inside a chip, and by adjusting the number of cores active inside a chip, according to the processing needs,

we lower the level of the static power due to the current leakage. Today's modern Unix-based OS's offer capability for making usage of such principle by dynamically adjusting the number of cores used in a multi-core chip, based of the CPU utilization, number of threads running in the system, or type and prediction of workloads to be run. E.g. in Linux systems, it is present a module for the implementation of the DPM strategy which is called *hotplug governor*. For this governor the decisions are primarly based on the CPU load, with changing to the highest frequency if the runtime tunable "up_threshold" is crossed. If on the other had the "down_threshold" is crossed the governor send the frequency to the next lower point from the frequency table. The main novelty of this governor is to set cores online and offline based on different sampling windows of the CPU utilization. In modern mobile devices where there is the presence of graphical processing unit (GPU), the management of graphic cores is also of great importance.

2.4.3 Hybrid strategy management

In these works [42, 43], authors explore the usage of both DVFS and DPM in the context of multi-core architectures for energy reduction. Their work is based mostly of the mobile architectures and the decision of ratio to use between DPM and DVFS is based on the type workload and on the level of which workloads are present.

2.5 Software hooks for energy management

In order to have a full potential from the strategy of power reduction, we need to explore possible actuators that reside on the software side.

2.5.1 Workload parallelization

With the unsustainable power dissipation of the previous unicore chips due to the continuous rise in clock frequency, there was on obvious move towards multicore design. In the multicore era, in order to exploit the full potential of many core chips the programming model should be oriented towards exposing the natural parallelism inside the software. In theory exposing software parallel units of execution, (e.g. threads) in the same level as the core availability in the chip can provide speedup in terms of execution time. In powerful machines its even demonstrated the exposing of the same number of threads as the CPU leads to an underutilized performance capabilities [44]. As authors show in [44], with a high number of cores, maximum benefits can be gained by exposing parallelism to a scale bigger than the number of available cores in the CPU. The level of parallelism is dependent of several factors connected with the structure of the software

but also sometimes with the input level. From the Amdahl's law by knowing the parallel fraction p of a software we can calculate the speedup of the parallelization with a factor n [45]. According to the research [37, 46, 47] this can impact not only the performance but also the energy efficiency.

2.5.2 Core type selection

From a technological point of view using dedicated cores for a specific types of tasks is the optimal choice in terms of efficiency, we get the most performance with the smallest power dissipation. Heterogeneous architectures give the possibility to have this matching between core type and task to be executed. In reality, there are many factors that limit this bright prospective of boosting efficiency. From the chip production point of view the heterogeneity level must be kept to a certain degree, because the more a core is specialized the smallest the range of application that could benefit from it is. From the runtime system prospective, matching the performance requirements with the level of efficiency is a difficult task knowing the variability of workloads that especially the mobile domain faces.

2.5.3 Load level

Since release of 2.6.23 Linux has a modular scheduling framework, where can be placed many scheduling classes which implement different scheduling policies. The linux scheduler has the task of allocating hardware execution units to system processes, in order to provide maximal performance and throughput while minimizing power dissipation. Perfect scheduling would require the prediction of future behavior of process submission in the system and also their characteristics. The kernel tracks down how much time a process spends executing, but there is a need to assess the impact of this time on the overall CPU engagement. The terminology used in the scheduler is *load*, which represents an instantaneous quantity of how much is the process loading the CPU (on the contrary of *usage*, which shows a cumulative property). Modern calculation of *load* includes assessing **per-entity load tracking (pelt)**. A process is contributing to the system load not only if is running but also if is ready to run. The load is calculated on 1ms scheduling periods and *pelt* is evaluated to include the contribution of a process not only in the current window of scheduling but also on the previous windows [48]. In current schedulers, the *load* includes the effect of processes which are not runnable but blocked, since they also contribute to the load of the system.

2.5.4 Summary

It is in the middle 2000s that power consumption has been recognized as a serious design constraint for chip manufacturers. While on mobile devices and embedded systems the focus is mostly on the energy consumed by the battery, in desktop systems the problem is often related to thermal issues. The dissipated power in the chip is so high that the normal cooling systems cannot maintain normal working temperatures inside the silicon. This was the reason for the change to the multi-core era, trying to continue in the performance advancement with a more controlled power and thermal budget.

In recent years the focus has shifted from the continuous race to achieve higher hardware performance, to power-aware computing systems where power and energy efficiency are of great paramount. Dynamic and static power dissipation in multi-core chips is the focus of the research community and industry. Hardware and software mechanisms have been explored in order to provide every time better levels of energy efficiency in a scenario where the number of computing devices is continuously and unstoppably rising.

Chapter 3

Utilization level scaling

Utilization level or *workload*, is a metric which is traditionally used by the scheduler to schedule tasks and map them on the CPU. Task mapping in a certain core influences its utilization factor and also the power usage of the core. The way "load" is measured is by accounting the utilization of all tasks mapped to the core, in order to have a number (as a percentage) that tells "how much the core has space left". This parameter could result beneficial for lowering the power usage of the CPU.

3.1 Real time schedulers

Pre-emptive operating systems are based on scheduling algorithms with priority metrics. Scheduling prescribes a runtime environment in which tasks, with a priority attribute, are dispatched in priority order. Priorities traditionally are, essentially, considered static. Tasks are either runnable, in which case they are held on a (priority ordered) run queue; delayed, in which case they are held on a notional (time-ordered) delay queue; or suspended, in which case they are awaiting an event which may be triggered externally (via an interrupt) or internally (from some other task). Most existing hard real-time systems are implemented using a static table-driven schedule. Priority-based scheduling has many advantages over this static approach ([49]). In essence, these advantages all relate to one theme: increased flexibility. However, in order to challenge the role of static scheduling as the premier implementation model, priority-based scheduling must:

- provides the same level of predictability
- account for different application characteristics
- have safe implementation candidates

3.2 Sched_deadline

SCHED_DEADLINE provides a scheduling policy for real-time applications by implementing a resource reservation algorithm in the Linux kernel. The algorithm is called CBS (Constant-Bandwidth Server) and is used for resource reservation in an EDF (Earliest Deadline First) scheduler [50]. Resource reservation mechanism is important in providing temporal isolation of the tasks in order to allow real-time scheduling capabilities in general-purpose operating systems. The technique is based on the idea of giving each task a 'reservation' with characteristics(Q_i, T_i), meaning that each task is guaranteed a run of maximum Q_i every period of T_i. If a task tries to run more than Q_i inside a period then the scheduler throttles the task until the end of the current period. So, each task is assured with some running time, with a mechanism to not let it exceed the reservation. CBS is an aperiodic server algorithm that implements CPU reservations differently from other algorithms in the group like Constant Utilization Server or Total Bandwidth Server [51]. In CBS, if the fraction of CPU time given to a task is U_s then its contribution to CPU utilization is no bigger than U_s, even in case of CPU overloading. The algorithm keeps track of two parameters for each task, such as q_i (the amount of time the task has left for the current reservation), and d_i (the scheduling deadline of the task). The last parameter is used to assign dynamic priority to the tasks. CBS operation is like this: when a task wakes up at time t the scheduler checks if the current deadline d_i can be used ($q_i < (d_i - t)\frac{Q_i}{T_i}$), otherwise a new deadline is generated $d_i = t + T_i$ and q_i is reinitialized to Q_i. While a task is running its remaining runtime is depleted with $dq_i = -dt$, until the runtime q_i becomes 0 in that moment the task is throttled and cannot be selected for execution. After going to the throttled state the task can exit that state when $t = d_i$, after which the remaining runtime is recharged to Q_i and the deadline is postponed to $d_i = d_i + T_i$, decreasing the priority of the task.

3.2.1 Control groups in Linux

One of the earliest partitioning mechanisms in the Linux kernel for process aggregation in terms of resource tracking are control groups(cgroups). A cgroup is able to associate a set of tasks with specific parameters, to a set of subsystems, which most of the time are resource controllers that put limits or schedules resources [52]. Cgroups provide many types of controllers.
cpu- This provides a guarantee for a minimum number of CPU slices when the system is busy.
cpuacct - Provides accounting for CPU usage from a group of processes [53].
cpuset - This binds a set of processes to a set of CPUs or NUMA nodes [52].
memory - This controller limits the process memory, kernel memory and

swap memory used by cgroups.
blkio - Control and limit access of cgroups to block devices.
perfevent- Allows access to performance counters related to a group of processes.

3.2.2 Multiprocessor scheduling

In multiprocessor systems, real-time scheduling techniques can be divided in two groups: partitioned and global. In partitioned scheduling, there are queues specific to a CPU, on the contrary in the global scheduling fashion, structures are shared among CPUs(cores). In Linux systems there is a mix of techniques, every CPU has its own run queue but the processes can move from one queue to the others. In sched_deadline there is also a run queue for each CPU and are provided operations for moving processes from one queue to the other. Cpuset is one mechanism to ensure that a process or group of processes to run on a specific CPU (or list of CPUs) and memory nodes. Another mechanism of Linux to select specific CPUs for execution is *processor affinity*. With Linux system call sched_affinity() it can be set the schedulability of a process or thread, to specific CPU(CPUs). With processor affinities it can be realized different scheduling schemes such as global partitioned or clustered scheduling [54].

3.2.3 User space API

The two main attributes of sched_deadline scheduler are: timing guarantees and enforcing timing confinement. The first one gives each task certainty of a given time for execution which the task will benefit from, while the second confines each task in its temporal interval protecting the other tasks in the system. For the API two new system calls have been used in the `SCHED_DEADLINE` class, which are `sched_setattr()` and `sched_getattr()`. The prototype of the systems calls is shown below:

Listing 3.1: Prototype of new system calls

```
#include <sched.h>
struct sched_attr {
        u32 size;
        u32 sched_policy;
        u64 sched_flags;
        /*SCHED_OTHER, SCHED_BATCH*/
        s32 sched_nice;
        /*SCHED_FIFO, SCHED_RR*/
        u32 sched_priority;
        /*SCHED_DEADLINE*/
        u64 sched_runtime;
```

```
        u64 sched_deadline;
        u64 sched_period;
};
int sched_setattr(pid_t pid,const struct sched_attr*
    attr, unsigned int flags);
int sched_getattr(pid_t pid,const struct sched_attr*
    attr,unsigned int size,unsigned int flags);
```

3.3 Controlling synthetic application utilization level

In controlling the utilization level of a CPU or core, there is a need for a repeatable approach of a load generator generating different kinds of loads on a multicore system that can be spread across cores using different policies. In our work for synthetically achieving different levels of utilization in MPSoCs, we used a load generator called Spurgbench [55]. With this approach, we maintain a specific *utilization level* according to the parameters that we give to the tool. Different utilization levels are used to stress hardware cores for energy measurements. The load generator has two responsibilities, 1) to direct control flow between the operation and the idle state and 2) to estimate the instantaneous cycle count of the operation. The state machine of the tool has three states: operation, sleep and estimate, and the transitions between these states are illustrated in Figure3.1. Spurg-

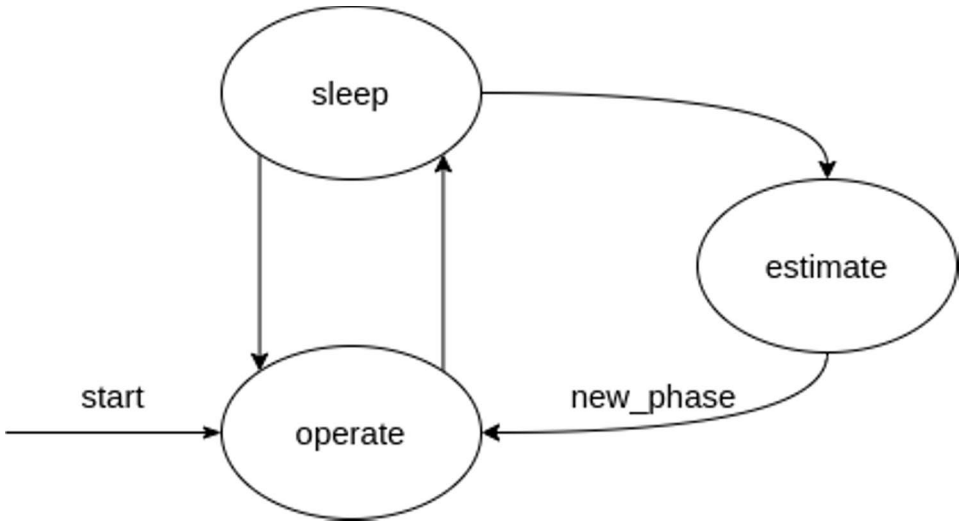

Figure 3.1: State machine of load generator operation

bench works in phases starting with phase 1 in state operation. On the

transition from estimate to operation the load-generator enters a new phase i + 1. Each phase has a set of parameters calculated in the estimate phase. At the beginning of phase i the load generator will transition between the operation and sleep states o(i) times, and then enter the estimation state. In reality, the tool uses a double loop construct such that $o(i) = n(i) * m(i)$. The inner loop executes $n(i)$ times and has the purpose of increasing the sleep time needed. In this manner is possible to have a sleep time longer than T_{run}. The outer loop executes the inner loop $m(i)$ times and is used to control the time between measurements. For further details on the utilization calculations refer to [55]. The load generated from the tool consists of operations. Different implementations of operations can be used with Spurgbench, but we searched for one that can exercise different parts of the hardware such as cores, ALUs, floating-point units, memory interfaces, memory hierarchy, and other units. The characteristic of load should be such that can be controlled in the utilization level and also adapt to the changing performance of the system. The execution time should be small enough to give the controller the granularity to adapt the utilization level in a fast way. In Listing3.2 there is a description of the operation used.

Listing 3.2: Operations used to generate load with Spurgbench

```
int operation()
{
        int i;
        double a = 2.0;
        for(i=0; i< 1000; i++) {
        a*=2.0;
        }
return 0;
}
```

3.4 Controlling real application utilization level

In our experiments for validating the energy efficiency model, we used a real-world application that comes from the Synchronous Dataflow (SDF) Models of Computation (MoCs) world. As an application case study we choose to use an SDF implementation of a stereo matching application from the computer vision application class [56]. The idea behind stereo matching is to compare input images taken by near cameras in order to produce as a result the depth of a scene. The algorithm used in this application offers a great opportunity for expressing parallelism which serves our approach based on parallel instances of an actor. The stereo matching application is

composed of 12 actors and two tunable parameters which can be used for having a high degree of parallelism of the most compute-intensive actors which are: *CostConstruction*, *AggregateCosts* and *ComputeWeights*. The parameters are *nbDisparity* and *nbOffsets*. The first represents the number of distinct values that can be found in the disparity map, while the second influences the size of the pixel area to be considered for the correlation calculus. We control the utilization level of the most intensive actor using `SCHED_DEDLINE` scheduler and adjusting the parameters of the period(T), runtime(q), and deadline(d).

3.5 Summary

This chapter explained the notion of *real-time schedulers*, together with a new scheduler class recently introduced into the Linux kernel. Controlling the scheduling of the software components is a crucial part of the aim of achieving better energy efficiency. Today new classes of schedulers and mechanisms in Linux systems, make it possible to implement efficient strategies for benefiting from multi-core heterogeneous platforms.

Chapter 4

Contribution of the Thesis

In this chapter are discussed the novel contributions of this research work on the energy-efficiency in multi-core heterogeneous platforms. New concepts such *platform configuration points*, or known techniques such as offloading, are used and explored in order to increase energy-efficiency in various use cases.

4.1 Energy Efficiency in heterogeneous systems

Energy today is the biggest challenge faced by computer systems for nearly every class of computer. First, power must be brought in and distributed around the chip, and modern microprocessors use hundreds of pins and multiple interconnect layers just for power and ground. Second, power is dissipated as heat and must be removed.

An important factor that chip designers and users need to consider is energy and energy efficiency. As explained in chapter two, power is simply energy per unit of time: 1 watt = 1 joule per second. There exists always the dilemma: which metric to consider for comparing processors: energy or power? In general, energy is always a better metric because it is tied to a specific task and the time required for that task. The energy to complete a workload equals the average power times the execution time for the workload.

One of the strategic objectives considered with the use of Multiprocessor System-on-Chip (MPSoC) is energy. Energy consumption is an intrinsic metric that is dependent on multiple factors, like the ratio of dynamic and static power dissipation, the execution time of the application, and physical plus architectural features. For defining a winning strategy for minimizing energy consumption, there is a need for a detailed analysis of the interplay of such factors and their impact on the objective. As the diversity of

workloads increases, the optimal choice for mapping software to hardware becomes a challenging problem. One of the recent techniques explored by researchers involves using heterogeneity in order to achieve better performance and energy results([57], [58], [59]). As explained in Chapter 2, there are in usage different types of heterogeneity, but in common is the intuition of using it as a mechanism to save energy. By matching the task with the appropriate hardware execution unit, the required result would be the task completion within the timing requirements with the least energy spent. There is a recent industry trend towards increasing the level of heterogeneity inside a chip for power-saving purposes ([60], [61]). Also, from ARM the latest technology in the matter named DynamIQ [1] provides many options for organizing high-performance and energy-efficient cores inside a cluster. Following this trend, there is a need to study more in detail the characterization of the power dissipation in multicore heterogeneous platforms using different systems actuators present at the OS or hardware level. Because the consideration of parameters such as type of core, frequency and core utilization rate affect the power dissipation, we are going to explore the impact of these arguments on the energy efficiency of heterogeneous MPSoC

4.2 Characterizing MPSoC power dissipation in heterogeneous systems

One of the earliest heterogeneous architectures designed for power efficiency is Big.LITTLE from ARM [62]. As explained in Chapter 2 it is a *single-ISA* heterogeneous architecture base on two types of cores, the one optimized for energy efficiency, and the one optimized for performance which is power hungry. In this type of platform, we introduce a new concept which we name *Platform Configuration Point*. A single configuration point is composed of the joint state of platform variables which affect the power dissipation of the platform. These variables are system knobs that can be modified at runtime in order to produce an effect on the resulting power that the hardware platform dissipates. As system knobs we focus on:

- Dynamic Voltage and Frequency Scaling (DVFS)
- type and number of cores to use in the heterogeneous system
- number of cores to use (level of parallelism)
- utilization level of the core (load level)

In order to better characterize the power of a heterogeneous platform, we explore the configuration points which provide high levels of energy efficiency for different levels of performance. The configuration analysis will be based on specific performance levels required by the application. Taking into account the number of DVFS levels for each core, the number of cores,

[1] http://developer.arm.com/technologies/dynamiq

and the utilization levels each cluster is able to provide, there is a large number of possible configurations to consider. Furthermore, with modern architectures, that aim to increase the level of heterogeneity, the number of configuration points will grow exponentially. As a parameter inside each configuration point, we use also the utilization level of the core running a specific task, as described in Chapter 3 the mechanism is based on a class of real-time scheduler called `SCHED_DEDLINE`. A well-known execution strategy of tasks for a CPU is called Race-to-Idle [63], where the logic is to use the highest possible frequency in order to finish the task as soon as possible, in order to minimize time. The reason behind this is the energy consumption minimization, but as shown in [64] this strategy is not working in modern MPSoC which have different power characteristics. In Heterogeneous Multi-core Processing (HMP) there is a different footprint of the power dissipation, in relation to the execution strategy. Recent studies ([65, 66]) show that achieving energy efficiency in heterogeneous architecture under a moderate load level on a core should be kept constant. In other words, we should try to keep the cores busy most of the time. A strategy coined with the name "never-to idle". Using the never-to-idle strategy scheduling heuristic, we search for the near-optimal energy efficient configuration between the number, type, frequency level of cores, and utilization level considering as a requirement the performance characteristic of the tasks to map. Using the utilization level as a scaling factor for changing the power behavior of the platform, brings our strategy close to the philosophy of the Never-to-Idle strategy which proves to be more energy efficient in HMP world.

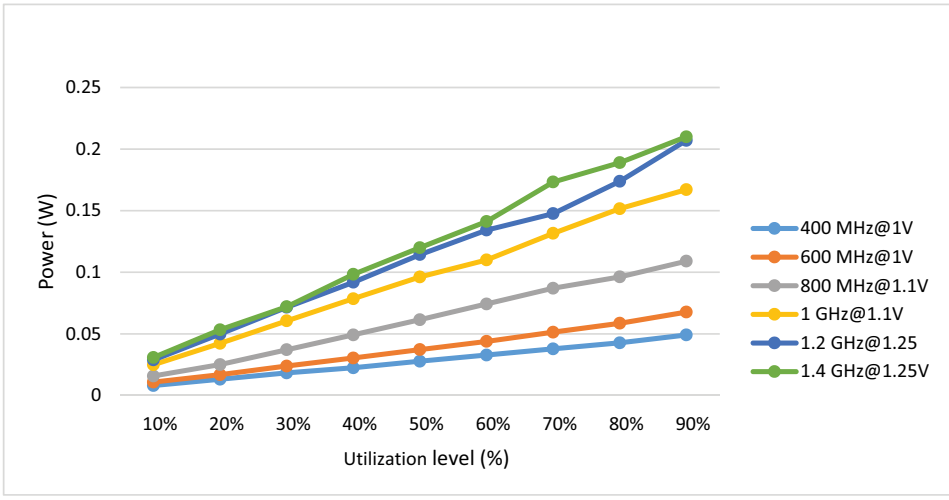

Figure 4.1: ARM Cortex-A7 core power dissipation for different utilization levels

In Figure 4.1 is drawn the graph that shows the relation between the utilization level of the core and the power dissipation, for different frequency levels. The test is performed with ARM Cortex-A7 energy efficient core architecture [62], which is used in some mobile multi-core chips like Exynos 5 Octa core [67]. The counterpart of the energy-efficient core is the one that is optimized for performance and consumes much more power than the first core type.

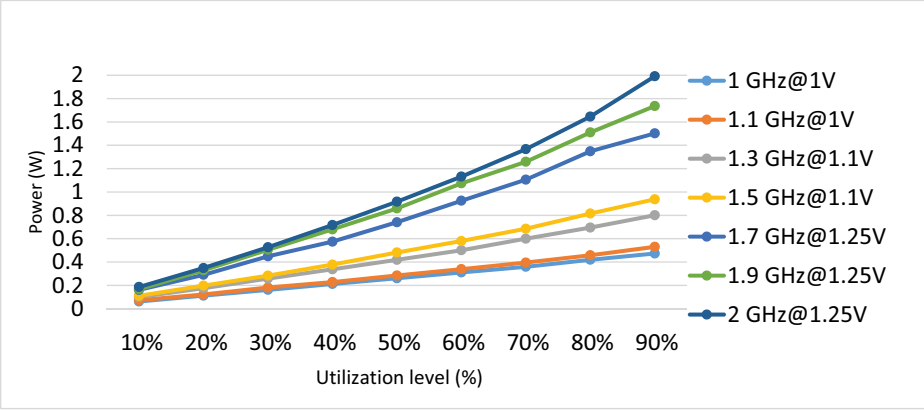

Figure 4.2: ARM Cortex-A15 core power dissipation for different utilization levels

In Figure 4.2 is shown the power dissipated by the ARM Cortex-A15 core, which is an architecture optimized for high performance in complex workloads. As noticed by the lines on the two graphs, the power increase is linear with the utilization rate. We experiment with loads from 10 to 90% explicitly trying to avoid the Race-to-Idle strategy of running at a 100% utilization level and full. As described in Chapter 2 there is a strong impact of the supply voltage in the power dissipation which can be noticed in the difference of going from 1.1GHz to 1.3GHz or from 1.5GHz to 1.7GHz where we see the difference in the supply voltage, from 1V to 1.1V or from 1.1V to 1.25V.

4.2.1 Energy Efficiency at the core level

As a workload for running in each core type, we choose the workload generator called Spurg_bench discussed in Chapter 3. The workload is composed of floating point operations designed to stress many parts of the core micro-architecture. The performance of core types is measured in operations/s achieved in different *Configuration Points*. From the power measurement data, as shown before, we can assess the energy efficiency of each core type under different levels of utilization and different frequencies. We express the

energy efficiency using the achieved number of operations per joule metric (ops/J). In Figure 4.3 are shown the energy efficiency figures for different frequencies of the ARM Cortex-A7 core type for different utilization levels. As noticed from the almost convex shape of the curves the highest level of efficiency is achieved for utilization around 60% - 70% and not the highest level present in the experiments which is 90%.

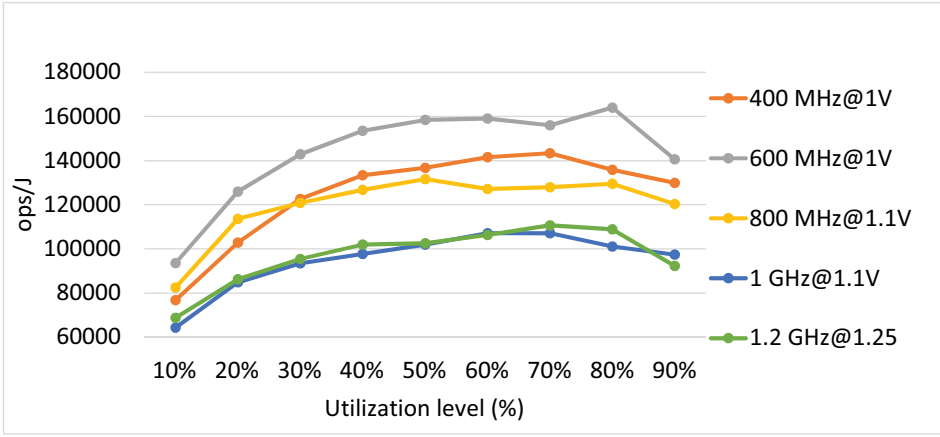

Figure 4.3: ARM Cortex-A7 core energy efficiency

The same conclusion can be derived from the graphs in Figure 4.4, where the ARM Cortex-A15 core shows the highest levels of efficiency around 60%-70%.

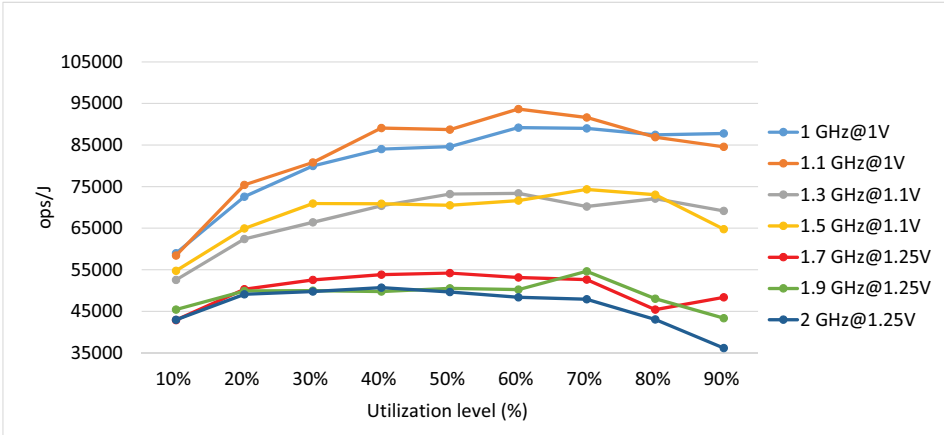

Figure 4.4: ARM Cortex-A15 core energy efficiency

4.2.2 Energy Efficiency Model

From the measurements conducted in the two core types before we can calculate the energy efficiency level of all configuration points derived from a platform powered by Exynos 5422 MPSoC. This chip contains two clusters of 4 cores with the A7 core composing the LITTLE cluster and A15 the big cluster. On each cluster, the DVFS is at the cluster level, and the cluster provides 1-4 cores available for execution.
From the energy model, we can plot the efficiency levels as a function of performance for all possible configuration points. Figure 4.5 shows the best and worst efficiency configurations for all reachable performance levels. From the Figure, we can observe that up to the performance level of $2e^{+}5$ operations per second, the density of configuration points is high and the difference between the efficiency of the worst and best configurations is large. Moreover, up to this performance level, there are configurations providing the highest possible efficiency level. After this performance level, the configuration density is decreasing and the highest possible efficiency levels are not reachable anymore.

Figure 4.5: Efficiency vs. Performance for all configuration points

4.2.3 Summary

In this section, we discussed the characterization of energy efficiency in heterogeneous MPSoC. We introduced the concept of platform configuration point and analyzed the achievable energy gains when exploiting at the same time heterogeneity, voltage and frequency scaling, and utilization rate con-

trol techniques.

We proposed an approach to build an energy efficiency model and derive a characterization of the platform based on platform configurations. We analyzed the energy efficiency variation for different platform configurations providing the same level of performance. We show that trading the number and type of core with frequency and voltage level and core utilization rate can lead to substantial energy gain.

4.3 Energy efficient scheduling in SDF applications

Achieving an energy-efficient application execution on multi-core heterogeneous architectures requires an overall view and deep understanding of the underlying software and hardware. First, the software needs to expose the right amount of parallelism in order to utilize the capabilities of the hardware, and then an efficient mapping of the software to hardware needs to be found. All of these objectives should be accomplished with respect to the application performance constraints. Different MoCs have been used for modelling applications; SDF [2], Cyclo-Static Dataflow (CSDF) [68], Kahn Process Network (KPN) [69]. One of the main purposes of these MoCs is to expose parallelism inside the application. In general, applications described by Dataflow Process Networks (DPNs) consist of a set of entities from which the application is composed, which are called actors. Actors communicate through FIFO buffers, and each actor can execute (fire) as soon as its input buffer contains enough data tokens (consumption rate). After execution, the actor produces a certain number (production rate) of data tokens in its output buffer(s). The popularity of these models comes from the simplicity of analyzing the application structure.

In dataflow MoCs an application is defined as a graph of concurrent tasks which communicate with each other through FIFO buffers. Synchronous Dataflow (SDF) [2] is probably the most used DPN, especially in signal processing applications. Inside a graph of actors, the production and consumption rates of the actors are fixed numbers, which enable a static analysis of the graph. Another valued property of SDF, not present in KPN, is the possibility to schedule the graph at compile time, as long as the graph is schedulable, deadlock-free, and consistent. In SDF actors are stateless, which practically means that an actor can start several replicas of itself in parallel if enough data tokens are available. With the exposed available parallelism, we can develop mapping techniques exploiting the benefits of using heterogeneous processors to achieve energy-efficient execution of applications. In order to enable the energy-efficient execution of a software application based on the concept of platform configuration points, we need

to change the scheduling and mapping of an application according to configuration points that provide high levels of efficiency. We choose from the model a configuration with a high level of energy efficiency and call this configuration point as Configuration of Interest (COI). To verify our approach, as a case study, we take the Stereo Match application discussed in Chapter 3. Our intention is to apply the COI on the *computerWeights* actor and change the default schedule provided by the tool `PREESM` [7].

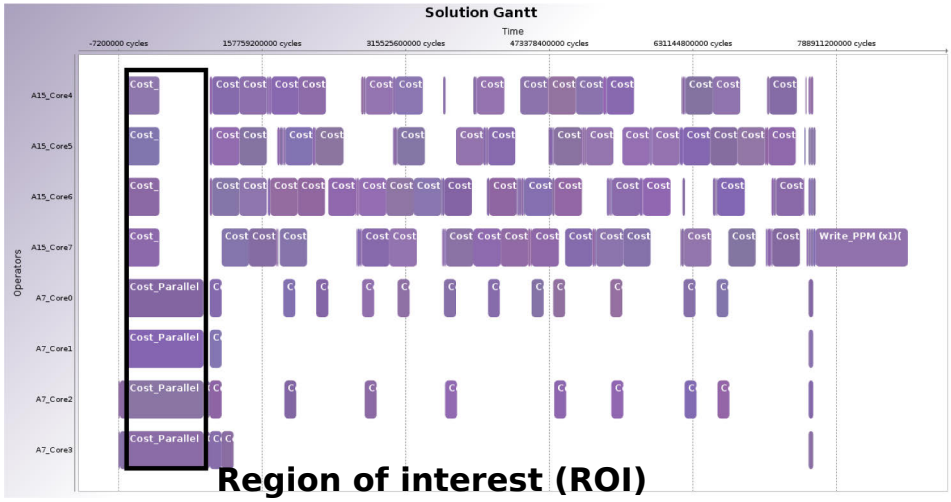

Figure 4.6: Gantt chart of the scheduling of actors in the stereo matching application

In the chosen COI there is a parallelism of eight, meaning that the two clusters of the big.LITTLE architecture are busy with each core running a thread that is a copy of the *computeWeights* actor. In Figure 4.6 is shown the modified schedule from the `PREESM` tool with the parallelism required from the selected actor. In the Figure, we have highlighted the Region Of Interest (ROI) in which we are interested. In that region, we will enforce the frequency and utilization level of each cluster. The configuration we are interested in is 60%/1.1 GHz/4A15/60%/200 MHz/4A7, which means that 4 threads will run in the big cluster at the utilization of 60% with 1.1GHz of frequency and 4 other threads will run on the A7 cluster with the utilization of 60% and frequency of 200MHz. We want to compare the energy spent during the ROI, with the default schedule which is run by the tool and means using the race-to-idle strategy. For the sake of comparison, we experiment with another configuration point that has lower efficiency than the COI. In Figure 4.7 we show the energy consumed for the ROI in 3 different configuration points:COI, race-to-idle configuration and another configuration with lower efficiency from the model. The results show that the COI

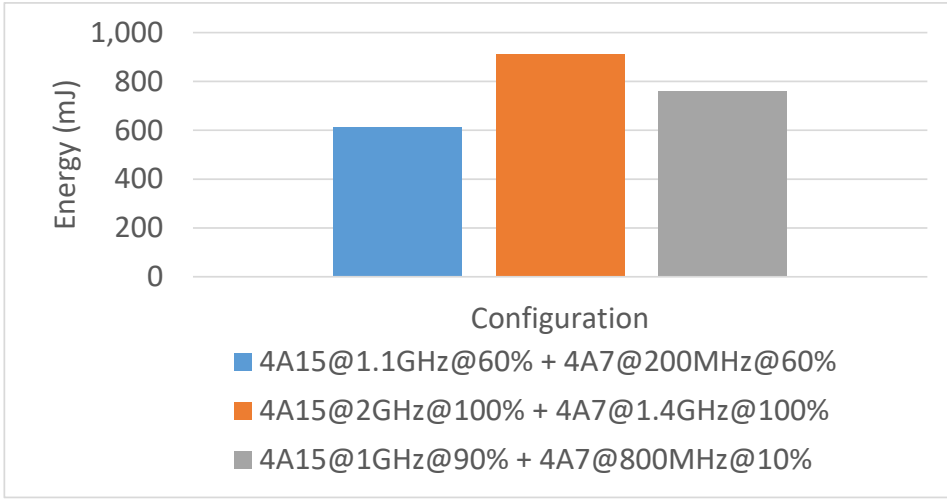

Figure 4.7: Energy comparison of different configuration points

achieves energy savings compared to the other strategies of execution.

4.3.1 Summary

In this subsection, we showed the applicability of the energy efficiency model in a real use case application. With the help of the SDF MoC we were able to obtain the level of parallelism suitable for the heterogeneous multi-core platform used in the experiments. By selecting the optimal configuration we showed that we're able to have up to 30% energy reduction for a single actor compared to the default strategy proposed by the tool. This model applied to the rest of the actors could provide even more energy benefits.

4.4 Power models through hardware performance counters

In Section 4.2 we discussed the characterization of the energy efficiency of heterogeneous MPSoC, where the concept of platform configuration point was introduced to set a level of performance and energy efficiency. In such a case the energy efficiency model was created by measuring the performance and power dissipated by the MPSoC in possible platform configuration points. High levels of heterogeneity recently introduced in embedded architectures, mentioned before in Section 4.2, yields an increase in the design space exploration to find efficient use of platform actuators. By increasing the number and type of cores, and also the granularity of the voltage islands, together with the number of voltages and frequency levels for each computing element, there is an increasing number of operating points on

which the platform may perform. In this scenario making the right choice for execution could have a tremendous impact on energy efficiency. Another important factor to consider in today's multi-core architectures is heat. The temperature of the chip has a major effect on the power dissipation of today's systems [70], which makes it an important factor to account for in order to make the optimal energy-efficient choice.

To manage efficiently the workload scenarios faced by mobile devices, edge devices in IoT, or nano data centers, there is a need to continuously monitor power data in order to choose the optimal power and performance trade-off. Unfortunately, most of the hardware platforms today are not equipped with power sensors, which significantly complicates energy-efficient management of the system settings. In the absence of a physical way to measure the power dissipated inside the chip, there is a need to predict the power dissipation related to running software. Furthermore, knowing the impact of static power on the overall chip dissipation figure, we need to account also temperature, in order to have what we call a thermally aware energy efficiency model. As studied by [71] power prediction based on Hardware Performance Counters (HPC) is a reliable way to estimate power dissipation. Experimenting with the big.LITTLE architecture with two clusters of A7 and A15, measuring the core power dissipation we can predict power based on the following formulas:

$$P_{\text{A15}} = \underbrace{(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk})}_{\text{dynamic activity}} + \underbrace{\beta_b V_{DD}^2 f_{clk}}_{\text{BG dynamic}} + \underbrace{f(V_{DD}, T)}_{\text{static}} \tag{2}$$

$$P_{\text{A7}} = \underbrace{(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk})}_{\text{dynamic activity}} + \underbrace{f(V_{DD}, f_{clk})}_{\text{static and BG dynamic}} \tag{3}$$

where N is the number of events selected, β_n is the weight given to a certain event, E_n is the number of events per second divided by the frequency (f_{clk}) in MHz, V_{DD} is the operating voltage and T is the temperature of the core. By defining the coefficients of the above formulas by using the regression method and a set of training workloads, we can also assess the performance of the application in terms of Instructions per Second (IPS). In this way, we can build the energy efficiency model based on platform configuration points that take into account the temperature of the chip. Sampling the HPC during the run of an application, the runtime system can estimate the power and react in case of an increase of the power dissipated, by changing the configuration point of the application running.

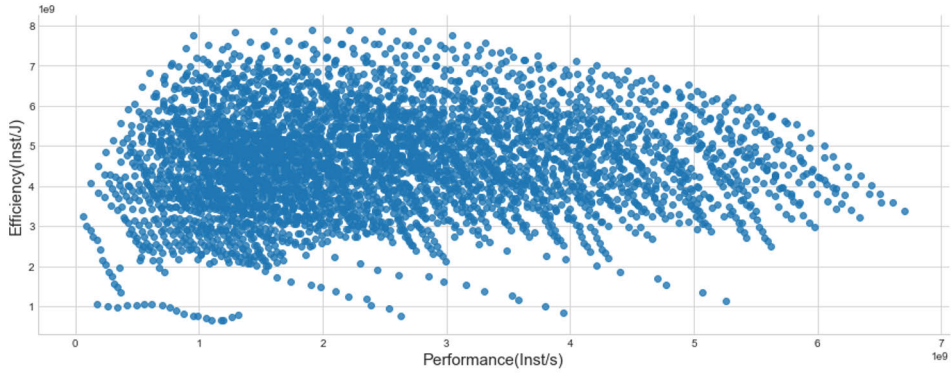

Figure 4.8: Energy efficiency of configuration points from the model

In Figure 4.8 are shown all platform configuration points set in the efficiency-performance plane. As we can notice, by going towards high levels of performance the density of the points decreases, which means that we have fewer options for configuring the running application. A useful scenario, for consulting the model is the case when the temperature changes during the application running.

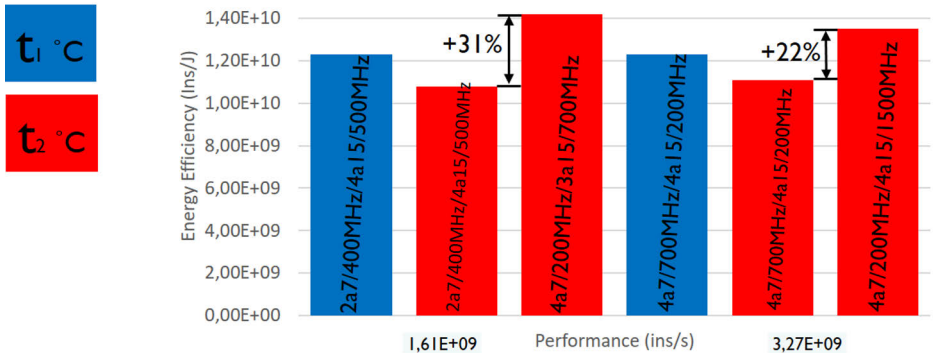

Figure 4.9: Application reconfiguration due to temperature change

By starting execution in temperature t_1 (Figure 4.9) the application has certain efficiency and performance, in case the temperature increases to t_2 then the configuration point drops in energy efficiency, that is why the application has to move to another configuration point which provides better efficiency in with the new thermal conditions. The same scenario can be found on the right part of Figure 4.9 with a higher starting performance requirement.

4.4.1 Summary

The continuously growing depth of heterogeneity in today's multi-core chips, plus the absence in the commercial MPSoCs of physical resistors for measuring the power dissipation in the hardware, brings a further need for estimating the power with a different approach. The usage of HPCs for power estimation is a proven technique that provides low levels of error in predicting the power. In characterizing the energy efficiency of the multi-core chip by configuration points, we remove the need for estimating the power in each of the possible configuration points. By sampling the counters in a configuration point and consulting the model we can estimate the efficiency of other configurations.

From the configuration points in the model, we estimate that only 1% of them represents the highest levels of efficiency. Including temperature in the power estimated is an additional value in the applicability of the model in cases of changing thermal conditions.

4.5 Reaching efficiency through mobile offloading

At the moment, human society is already in the era of the Internet of Everything(IoT). The usage of IoT, based on embedded devices is rapidly growing. The report from Global System for Mobile Communications Assembly (GSMA) [72] has shown that the total number of connected devices is estimated to reach 24.6 billion by 2025. This growth is fueled mainly by the increase in different application types populating our digital life. Smart terminals, smart voice assistants, and autonomous driving are a few examples of applications that will increase dramatically the number of connected devices and also the amount of data produced. On top of this development is also artificial intelligence technology which is fueled by the vast amount of data being produced. In this new ecosystem, challenging requirements are set for future networks in terms of bandwidth, latency, reliability, and energy efficiency. Edge computing can theoretically provide high bandwidth, low latency, and the computing agility required by today's new digital services. Edge computing [73] refers to a concept in which a distributed architecture decomposes the large-scale computing of the central node into smaller and easier-to-manage parts, and then moves them to edge nodes for processing. The edge nodes are geographically located close to terminal devices and have higher transmission speeds and low latency. The edge refers to the base station or server close to the user side. One key aspect of the edge is that many of the devices may be battery-powered or deployments are power limited. The more energy efficient the processing is done at the edge, the more computation can be done with the same power budget. Application

complexity and power consumption are increased when distributed AI is deployed to edge devices. Moreover, the increase in the number of devices produces a significant rise in the energy consumed by the technology [74], thus playing a concern in future strategies for curbing energy consumption. There are different strategies to lower power dissipation in MPSoC and consequently the energy consumed. Yet, frequently targeting only the MPSoC is not enough in reducing energy consumption, since other parts of the platform might consume more energy [75]. Depending on the platform type, I/O connected, and the type of applications executed on it, different execution strategies might be the solution to energy conservation. For instance, the Race to Halt strategy is proved to be a solution if the CPU is not the major power consumer in the platform [76]. By contrast, if the static power dissipation of the platform is relatively low, the execution with a lower clock frequency of the application might save energy by going slowly to the application results [76]. In order to have a holistic approach, we need to find a working configuration of the platform where the energy consumed is minimized.

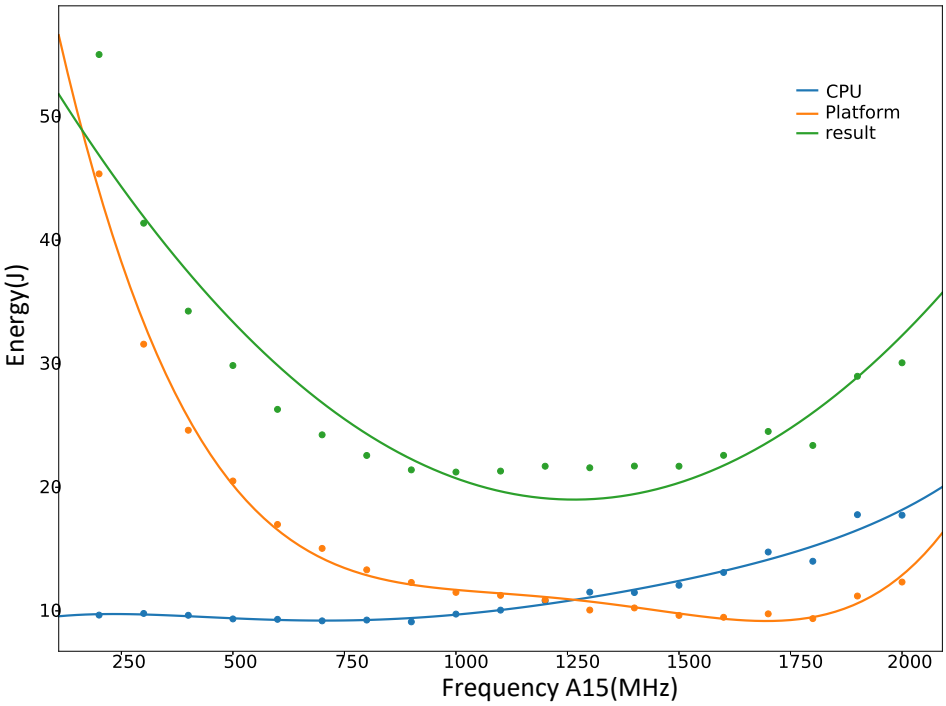

Figure 4.10: Energy measurement, at CPU and platform level

In Figure 4.10, are shown the energy measurements of the experiments of running an AI neural network in *single board computer* (SBC). The exper-

iment is conducted in different CPU frequencies and the energy is measured at the CPU and platform level. The resultant energy consumed is shown in the green line. From this graph, we see that in this case, the winning strategy for energy minimization is having a middle frequency for minimizing the overall platform energy consumption.

4.5.1 Computational Offloading

In the present time, we are witnessing a change in user computing preferences toward mobile computing technologies. Cloud services offered for mobile platforms are increasing in number. Also, the number of smartphones used is going to increase shortly soon with the usage of different services like e-learning, gaming, entertainment, and health care. AI technology is continuously increasing its presence in applications running on mobile devices in different sectors like natural language processing (NLP), entertainment, working, etc. Even though continuous technological and performance improvements, embedded mobile computing devices come with physical constraints, such as processing power and, sometimes more importantly, battery life. Also, many AI applications are challenging in terms of computational requirements.

In the case where local resources are inadequate for on-device inference, computation offloading is an encouraging technique that offloads the inference tasks from end devices to edge servers. The design principle of edge inference via computation offloading is explained in Figure 4.11.

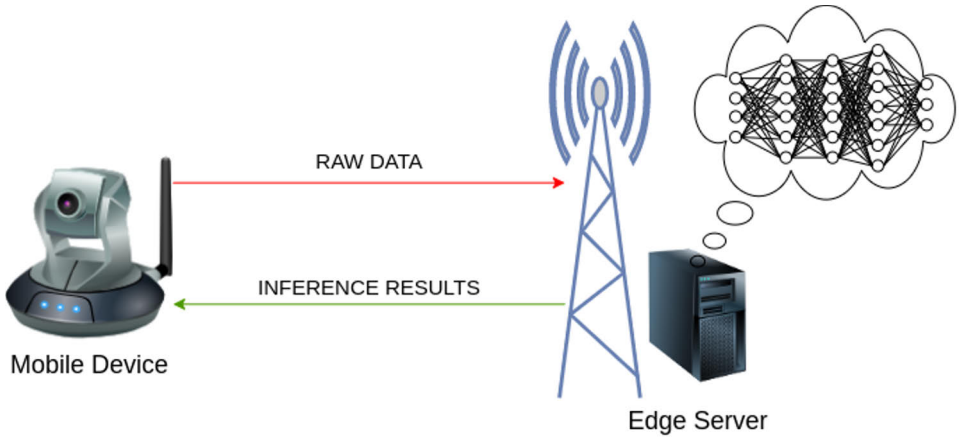

Figure 4.11: Edge inference computational offloading

The mobile platform transfers the acquired data for remote inference in case the available connection bandwidth is adequate and the required energy for local computation is high. If the total platform power to compute task

w is p_l, then the energy consumed by executing the task locally is $p_l * \frac{w}{s_l}$, where s_l is the speed of local computations. Otherwise with p_i the power of the system while idling and p_t the power of the platform while transmitting, the energy to offload the task can be defined as:

$$p_t * \left(\frac{d_i + d_o}{B} + 2 * t_{tcp} \right) + p_i * \frac{w}{s_r}$$

where d_i and d_o are data to send and receive from a remote site, B is the bandwidth of the connection and s_r is the speed of the remote site in executing the task. In this case, the offloading mechanism saves energy if:

$$p_l * \frac{w}{s_l} > p_t * \left(\frac{d_i + d_o}{B} + 2 * t_{tcp} \right) + p_i * \frac{w}{s_r} \tag{4.1}$$

Depending on parameters like B or d_o, d_i, p_l, p_t the energy-efficient solution might vary. Also, the model used for the AI application might make a difference in the energy-efficient choice. Performing object detection in a mobile platform is computationally expensive and energy demanding. We experimented by running an object detection algorithm such as Single Shot Detector (SSD), locally in a mobile platform or offloaded to a remote server. In Figure 4.12 graph c), is plotted a comparison of the performance vs. energy efficiency reached by the local platform and the remote side. We can notice that the mobile platform can achieve performance up to 1.2fps, and if we need an additional frame rate, the computation should be offloaded to the remote side. While from Figure 4.12 graph a), we show that offloading is energy efficient if the bandwidth of network transmission is as high as possible. On the other hand from graph b) we see that local computation is energy efficient if we use middle core frequencies instead of high or low frequencies.

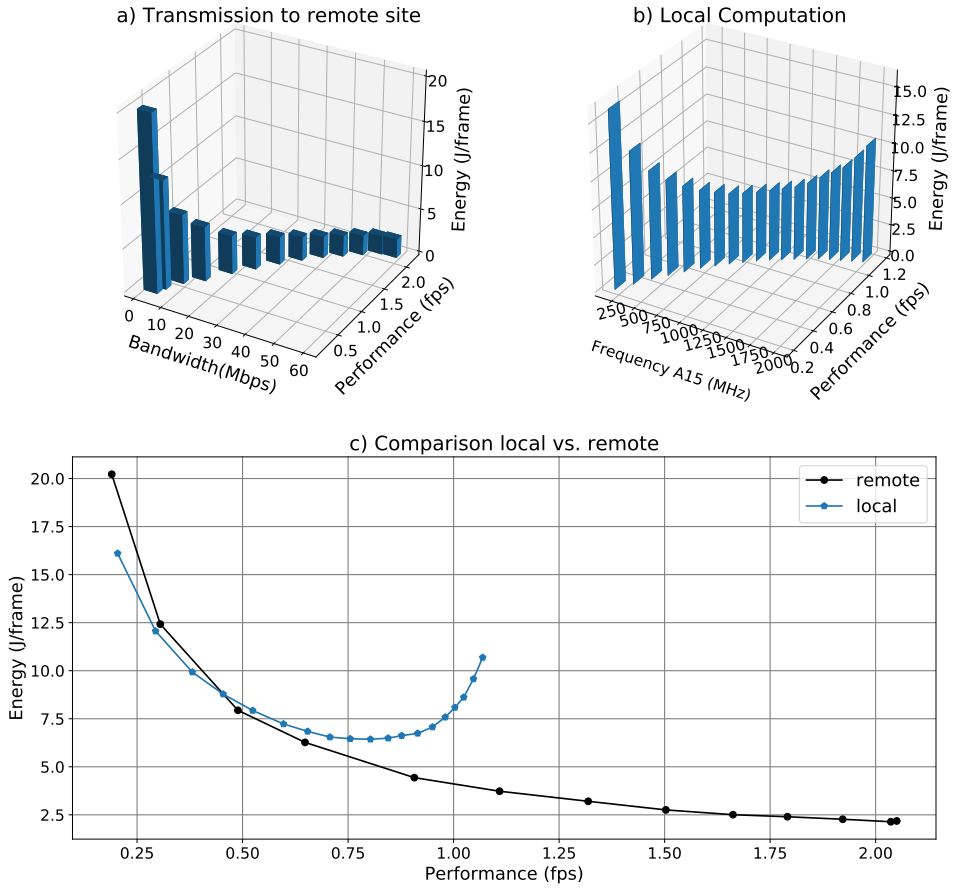

Figure 4.12: Energy/frame of a mobile platform versus remote computation efficiency

We experimented by running AI models for image recognition in a mobile platform equipped with Exynos 5422 chip, which is a big.LITTLE architecture with two clusters ARM Cortex-A7 and ARM Cortex-A15. Having at disposal 8 cores organized in two clusters and many DVFS points, configuration points are numerous. By running AI models in many configuration points we noticed that only by using A15 cores we were able to achieve performance, while A7 cores have an insignificant impact on running AI models.

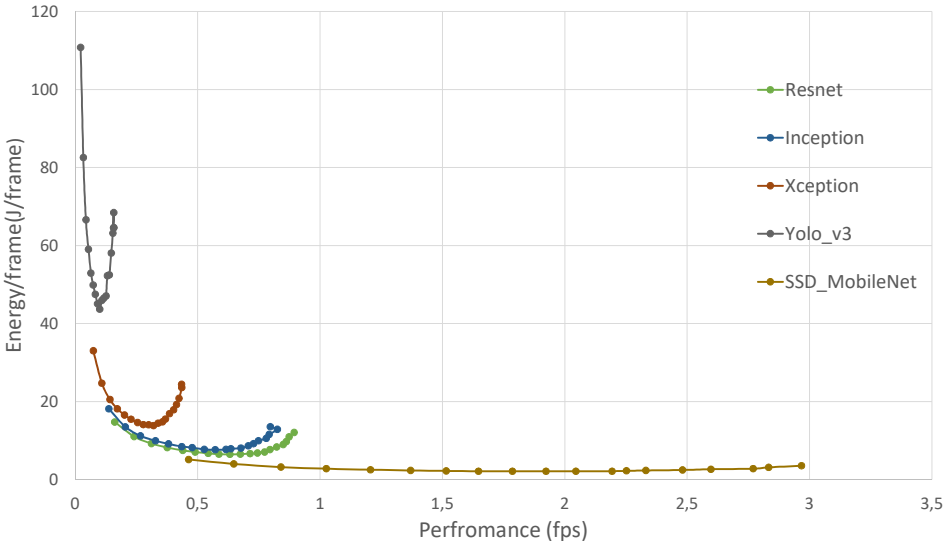

Figure 4.13: Energy efficiency of object detection models in the ODROID platform

We run five different models in platform configuration points that use A15 cores and plotted the values of energy/frame achieved by the models in different configuration points. The results are shown in Figure 4.13 where we see that the mobile platform achieves different energy efficiency results for different models. The worst energy efficiency figures are shown by YOLO_v3 which needs the highest energy/frame for performing detection. The lowest level of energy/frame is shown by SSD_MobileNet model which is the most energy-efficient of what we measured, and also gives a better performance range that can be achieved by the mobile platform.

4.5.2 Summary

In the search for an energy-efficient execution strategy, is not sufficient to focus only on the MPSoC of the platform to find optimal configurations in terms of energy conservation. In the presence of a digital data explosion, many components of the mobile platform need to be taken into account when searching for the optimal execution strategy. In this context computational offloading is a promising technique able to improve energy conservation while still achieving the performance requirements of many new applications. In the decision for offloading, particularly the bandwidth of the communication with the remote server is crucial for deciding to offload execution to the edge server. If the bandwidth is high enough then offloading becomes an efficient approach to save energy. Also with AI applications in embedded environments, not all platform configurations are acceptable

to run the algorithm. A large amount of performance is needed in order to run such an algorithm, and also trading off a bit of accuracy for more energy efficiency is very important in the mobile world.

4.6 Cloud-based telemetry data acquisition for improving processing platform efficiency

The upsurge in data produced in the digital environment fuels the rapid growth of AI applications, which need data to train and refine predictions. Mobile devices and IoT are on top of this data explosion, with CISCO predicting 85 Zettabytes generated by people, machines, and IoT in 2021, which is 4 times more than data managed in the cloud data centers. New applications are putting increased pressure on requirements like latency, privacy, security, reliability, and power dissipation. In this scenario, many AI engines are set to be closer to the data sources like in network edges. Also, major cloud providers like AWS, Azure, Google Cloud, and IBM Cloud are increasing the services they offer for IoT applications often with support for edge capabilities. Greengrass is a software product offered by Amazon that provides local computing, messaging, data management, sync, and ML inference capabilities to edge devices. Azure IoT Edge, from Microsoft, provides the ability to run AI, Azure and third-party services, and custom business logic on edge devices using standard containers. Google has designed the edge TPU (tensor processing unit) for edge inference, which has a smaller size and lowers power consumption than cloud TPU. Nvidia has designed Jetson TX2 for efficient AI computing. These devices make the inference process at edge servers more applicable. The need for edge devices comes mostly from the fact that in certain scenarios predictability is cardinal to the success of the application. E.g. in an Industrial edge system, the need for determinism of the response time is crucial to the well functioning of the work chain. Moreover, there is a requirement for the operational automation of factories and other industrial environments, which can be done with data analytics and machine-learning-based AI technologies. These use cases deploy edge systems for real-time control of the operations. The collection of large amounts of data is required from different system components like applications, edge platforms, and networks. Single-sourced and static data acquisition cannot meet these data requirements, which need to acquire information for different components in the production chain. It is therefore desirable to have a framework that integrates multiple telemetry approaches from different components. The telemetry framework brings a solution to this problem. The framework is composed of two parts as can be seen in Figure 4.14. On the left side, we have the edge node which is a highly

heterogeneous platform capable of handling AI applications with Convolutional Neural Network (CNN) for performing real-time inference. An agent is present on the edge node with the requirement of collecting telemetry from several components like application, hardware platform, and network connection. Once collected these metrics are sent to the right part of the framework, which is composed by the cloud side. On the cloud end these data are ingested and processed for immediate actions in case of tuning the edge or the data are stored for later analytics and machine learning.

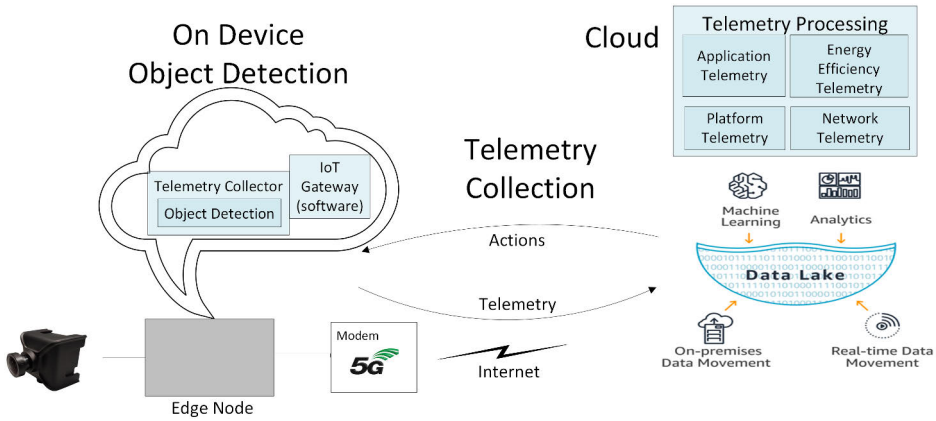

Figure 4.14: Telemetry framework

Selecting the right edge platform technology depending on the use case is paramount for achieving important requirements like performance, latency, and energy efficiency. Two well know edge platform competitors are systems based on GPU and those based on reconfigurable hardware (FPGA). In many cases, FPGAs are praised for their fitness in some application use cases which deal with AI inference. To check these assumptions on the achievable latency and performance of FPGA platforms for CNN-based edge applications, we evaluate two different platforms for the role of the edge node: an Nvidia Jetson AGX Xavier (as a representative GPU-based platform) and a Xilinx ZCU102 (as a representative FPGA-based platform). The Xavier is an embedded GPU platform that promise to offer high compute density and good energy efficiency for AI-related applications. The Xavier is equipped with 512 CUDA cores with Volta architecture GPU running at 1.37GHz and a 16GB LPDDR4X @ 2133MHz memory with a bandwidth of 137 GB/s, and a flash storage eMMC 32GB. The Xilinx Zynq UltraScale+ MPSoC ZCU102 board has a 16nm XCZU9EG FPGA, an onboard 4GB 64bit DDR4 RAM with a peak bandwidth of 136Gb/s. We target to check AI inference capabilities, and the power dissipation of the two platforms

while running neural network algorithms. The experiments are conducted using YOLOv3 and SSDResnet50Fpn algorithms for object detection which perform inference on a 420p video file. In Figure 4.15 we report the measurements done for both platforms for metrics such as end-to-end delay (EE latency) to process a single frame and the number of frames per second processed for a single dissipated watt (FPS/Watt) while running the two AI models mentioned before. The neural network is fed with the same video file and the power is measured on the entire platform. FPGA architecture is able to achieve good latency in time-sensitive jobs due to the circuit-level customizations on its massively parallel computing units.

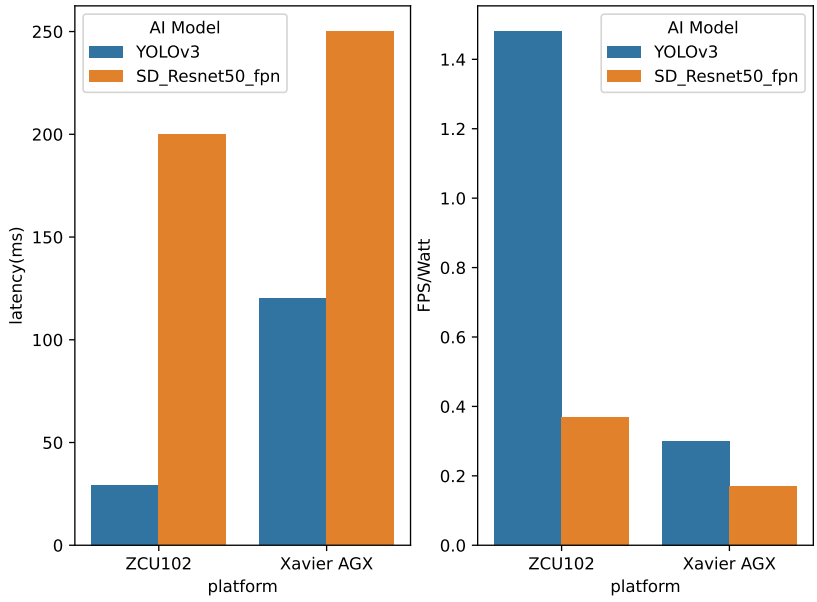

Figure 4.15: Edge platform technology comparison

From Figure 4.15 we notice that there is a clear advantage of the FPGAs platforms versus GPUs to be used especially in streaming applications, this is noticeable in terms of latency and energy efficiency. The SSD_Resnet_50_FPN is a heavier model compared to YOLO, requiring 178.4 Gops compared to 65.63 Gops of the other side.

Nowadays, FPGA platforms are widely used in industries ranging from telecom to aerospace. Its advantage over ASIC, in terms of design change, even after the product has been deployed in the field, has expanded the use of FPGAs. This feature allows the designer to upgrade from a remote

location eliminating the need for fabrication from scratch.

Another factor driving the growth of the FPGA market is the demand for extensive computation in applications which gives a new direction to the FPGA industry. Moreover, cloud computing and data processing in data centers have emerged as significant areas of application for FPGAs.

The diversity of Deep Neural Network (DNN) is also reflected in how much parallelism is available at each level mentioned earlier. Thus, any fixed hardware architecture that instantiates a fixed number of parallel compute elements communicating in a fixed fashion has limitations on how efficiently it can execute a DNN. Especially considering the rapidly-advancing techniques for creating efficient DNNs, adaptability is key to staying efficient in the changing DNN inference landscape. In this context, the key advantage of FPGAs is the adaptable, fine-grained, massively parallel nature of the computing and memory resources offered. For example, the Xilinx FPGA devices facilitate a wide range of Deep Learning Processing Unit (DPU) architectures that can take advantage of the multiple levels of parallelism and can tailor to the specific requirements of a given DNN topology and the application-specific design constraints.

FPGA-embedded platforms are enabling applications where differentiation is key, power efficiency is critical, systems must be extremely responsive, and the latest algorithms and sensors need to be quickly deployed. With such platforms, the need for fast and automated decisions is even more emphasized, which gives greater importance to the collection of a broad telemetry range. E.g. the continuous monitoring of the AI model fitness in the edge platform could make straightforward, the retraining of the current model in the cloud and updating the version at the edge, or the deployment of another model which fits better on the current platform. The energy efficiency of the application and platform is of great importance in certain use cases, in such scenarios, the power dissipation of the platform is constantly monitored and the application performance could be adjusted in order to fulfill the power requirements.

4.6.1 Summary

The large compute and storage requirements associated with DNN deployment necessitate acceleration. Furthermore, different constraints might be imposed on the accuracy, cost, power, model size, throughput, and latency depending on the use case. Real-time and safety-critical applications such as augmented reality, drone control, and autonomous driving are not suitable for offloading to the cloud due to low-latency requirements and data transmission overhead. In cloud-computing and ML-as-a-Service contexts, data centers face ever-increasing throughput requirements to process astronomical scales of data [77], bringing additional challenges in energy efficiency

to minimize operating expenses. While cloud service latency is less critical compared to embedded scenarios, it still translates directly into customer experience for interactive applications.

In this context, the cooperation edge/cloud is vital in assuring certain requirements in energy efficiency, power, latency, and performance. The holistic telemetry collection becomes crucial in the process of providing real-time control of the different components in the chain of application execution. The telemetry collection and utilization framework provides a solution to this challenge by bringing together the best of the edge and cloud worlds.

Chapter 5

Overview of Original Publications

In this chapter is presented an overview of the original publications. Each publication is described in the main problem to research, and the results of the publication. At the end of each publication, is mentioned the author contribution. At the end of the chapter is presented the connection of the papers and their relation to the main research question.

5.1 Paper I: Core Level Utilization for Achieving Energy Efficiency in Heterogeneous Systems

The paper focuses on the issue of energy budget which is becoming a constraint in all computing systems. From mobile devices to supercomputers, the focus has shifted from performance to energy and power efficiency. Design metrics are not anymore solely based on performance, as the energy efficiency of application executions is becoming a predominant design requirement. In addition to established voltage and frequency scaling techniques, several semiconductor chip manufacturers introduced heterogeneous multi-core processors to increase the level of energy efficiency. The usage of this heterogeneity is complicated by the scheduling and mapping decisions needed to be made at runtime for application execution. The intuition behind this work is that in order to exploit the full potential of such architectures we need to make the right decisions because parameters such as type of core, frequency, and utilization usually affect the power dissipation and performance.

This paper analyses achievable energy gains when exploiting core-level utilization in addition to other control techniques such as heterogeneity,

voltage, and frequency scaling. We build an energy efficiency model based on platform configurations defined by core types, the different voltage, and frequency levels, and the core utilization rate. Based on the built model, we analyze the energy efficiency variations for different platform configurations providing the same level of performance. We show that trading the number and type of core with frequency and voltage level and core utilization rate can lead to substantial energy efficiency gains.
Author's contribution: The idea of using hardware and software actuators to control the energy efficiency of modern multi-core chips and the notion of platform configuration point was developed together with the co-authors. The experimental environment and the model building were set up by the author. Results and summary of the model prediction were discussed and defined with the co-authors. The author contributed to the write-up of the paper and the presentation.

5.2 Paper II: Exploring Energy Efficiency Model Generalization on Multicore Embedded Platforms

In this paper, we investigate the relationship between the energy efficiency model and the type of workload executed in modern embedded architectures. Following the work done in Paper I, from the energy efficiency model obtained, we select a few configuration points to verify that the prediction in terms of relative energy efficiency is maintained through different workload scenarios. As workloads, we use a combination of synthetic generators and real-world applications from the embedded domain. In our experiments, we use two different architectures for testing the model generality, which provides examples of real systems. First, we have a comparison of the efficiency obtained by the two architecturally different chips (ARM and INTEL) in different configuration points and different workload scenarios. Second, we try to explain the different results through the thermal management done by the two different chips. As a result, we show that only in the case of workloads highly composed of integer instructions the two architectures converge in the same direction. In the case of workloads with high levels of integer instructions, we need another model to find high-energy efficiency configuration points. Another observation that we made by the results obtained from these experiments, is the limited efficacy of synthetic benchmarks when assessing proper energy efficiency comparison. This can be explained by the fact that static power effects are not strongly present while executing synthetic benchmarks with only certain types of execution units stressed, which explains the different energy efficiency results between synthetic and real workloads.

Author's contribution: Continuing from the previous work the author and co-author decided to explore the model generality at the level of different types of workloads and also different chip architectures. The author proposed to explore the model usage in INTEL and ARM as main contenders. The experimentation infrastructure and implementation were completed by the author. The author and co-author discussed the results and formulated the conclusions. The author worked on the paper writing and presented the paper.

5.3 Paper III: Energy-Efficient Actor Execution for SDF Application on Heterogeneous Architectures

In dataflow MoCs an application is defined as a graph of concurrent tasks which communicate with each other through FIFO buffers. Synchronous Dataflow (SDF) [2] is probably the most used DPN, especially in signal processing applications. Inside a graph of actors, the production and consumption rates of the actors are fixed numbers, which enable a static analysis of the graph. Another valued property of SDF, not present in KPN, is the possibility to schedule the graph at compile time, as long as the graph is schedulable, deadlock-free, and consistent.
In SDF actors are stateless, which practically means that an actor can start several replicas of itself in parallel if enough data tokens are available. With the exposed available parallelism, we can develop mapping techniques exploiting the benefits of using heterogeneous processors to achieve energy-efficient execution of applications.
In this paper, we demonstrate a mapping technique to achieve energy-efficient application execution, integrated into the workflow of the Parallel and Real-time Embedded Executives Scheduling Method (Preesm) tool. Preesm is a development framework for the rapid prototyping of dataflow applications. Applications developed with Preesm target execution in heterogeneous MPSoC, where the proposed approach defines the optimal platform configuration to run an actor under defined performance constraints. We define platform configurations by the following set of parameters: level of parallelism, number of cores, type of cores, DVFS, and utilization rate.
Through experimental results, we show that the highest platform configuration point in the energy efficiency table provides an energy reduction of more than 30%, compared to the standard schedule proposed by the tool, within a single actor execution. By using a second platform configuration point from the energy efficiency table we show that the relative efficiency is preserved for the actor.

Author's contribution: The purpose of the work was to check the

applicability of the energy efficiency model inside a real tool used for developing and prototyping SDF applications. The author and co-authors suggested the idea of integrating the model in a real-world application and the author run experimental evaluations for energy savings. The paper was written with the contribution of all authors.

5.4 Paper IV: Energy Efficiency Platform Characterization for Heterogeneous Multicore Architectures

Runtime estimation of power dissipation and performance is crucial in every computing platform. In mobile systems, a special focus is set on energy efficiency in order to achieve the longest possible battery life and at the same time adhere to performance requirements. Powered by heterogeneous MPSoC's, mobile systems are called to reach an energy-efficient state of execution, with a runtime system or scheduler that requires knowledge of the current performance and power dissipation. Today, highly heterogeneous architectures provide many actuators to reach better efficiency, the effect of which is usually unknown at runtime.

In this paper, we propose a fast approach to build an energy efficiency model based on hardware performance counters. Our approach obviates the need for power sensors present at the chip level and deals with high numbers of execution modes. In building the energy efficiency model we account for the change in temperature which, as we show, has an impact on the optimal energy efficiency choice. The proposed approach reduces significantly the time to characterize the energy efficiency of a MPSoC and includes the environment temperature as a variable in determining the energy efficiency. The novelty of this approach compared to previous works is that it doesn't necessarily need power sensors for measuring the power dissipation in each configuration point, but by sampling the counters on one configuration point we can characterize the efficiency of other configuration points. From all the points in the model, we show that less than 1% of them represent the highest levels of energy efficiency possible, in all the performance spectrum's offered by the platform. Also, we include the environment temperature as a variable for defining the need for application reconfiguration. As we show by the experiment if the temperature changes, by re-configuring the application execution we can gain up to 33% in terms of energy efficiency.

Author's contribution: The author's idea of extending the energy efficiency model to make use of Program Counters in the estimation of the instant power dissipation is explored in this paper. Both authors contributed to the clarification of the structure of the paper and to the review of the paper. The author conducted the experimental part, which was later

interpreted by the two authors.

5.5 Paper V: Towards very low-power mobile terminals through optimized computational offloading

Energy consumption is a major issue for modern embedded mobile computing platforms, and with new technological developments, such as IoT and Edge/Fog computing, the number of connected embedded mobile computing systems is rapidly increasing. Heterogeneous multi-core CPUs seek to improve the performance of these platforms, with a particular focus on energy efficiency. By using different techniques like DVFS, core mapping, and multi-threading, a substantial improvement in the achievable CPU energy efficiency level for MPSoC can be observed. However, controlling only the CPU power dissipation has a limited effect on the overall platform energy consumption. Other components of the platform, including memory, disk, and other peripherals, play an important role in the energy efficiency of the platform and need to be taken into account. The availability of different sleep strategies at various levels of the platform makes the energy efficiency issue even more complex.

In this paper, we set the view of energy efficiency at the entire platform level and discuss computation offloading as a mechanism to help in reaching the optimal platform energy-efficient state. As an application, we consider object detection performed on several types of images to define when offloading is beneficial to the platform's energy efficiency. We survey the energy efficiency of different neural network algorithms in an embedded environment, with the possibility to perform computation offloading. Computational offloading is the mechanism of moving heavy tasks to more powerful computing units. This mechanism can be a winning strategy for achieving good levels of energy efficiency in the case of highly demanding applications.
We concluded that if the bandwidth of the network connection is large enough, then the offloading strategy turns out to be more energy-efficient than local computing. We surveyed the energy efficiency of different neural network algorithms in an embedded environment and concluded that not many neural networks for object detection can be handled by average embedded platforms. In some cases, to improve the accuracy between 20% to 30%, the cost in degrading energy efficiency is 170% to 180%.

Author's contribution: The main idea of the paper comes from the author, which extends the previous notion of configuration point to the entire platform, taking into account this time the whole platform power. Both authors discussed the idea of using offloading as an alternative to

local execution. The author set up a range of experiments and executed them. The interpretation of the results is done by the author. The paper writing was done in collaboration with the other authors.

5.6 Paper VI: Data Collection and Utilization Framework for Edge AI Applications

Edge systems are the deterministic embedded communication and real-time control engines that reside at the edge of the network and are closest to the physical world of factories and other industrial environments, e.g., motion controllers, protection relays, programmable logic controllers, and similar systems. Clock frequencies in gigahertz, larger memory sizes, higher numbers of input/output ports, and the latest encryption engines might seem to offer solutions for future requirements that are as yet unknown. However, when dealing with the timescale of industrial equipment, which has critical subsystems that operate on a scale of hundreds of microseconds (or less) but need to operate in factories and remote locations for decades, relying solely on a cutting-edge multicore embedded processor to scale in the Industrial Internet of Things (IIoT) space is risky. It could result in a short-sighted catalyst leading to a series of difficult and costly marketing and engineering trade-offs focused on managing functional timing issues stemming from performance bottlenecks. A much higher degree of freedom in scaling is desperately needed at the IIoT edge due to the timescales involved. Such scaling freedom can be unlocked by using programmable hardware that augments the software running on the embedded processor cores. The assumed scenario for this work is the following: an industrial system (it could be for example a patrolling robot or a manufacturing conveyor) is streaming live a video over a 5G network and requests as a service detected objects from the video stream. The object detection service is executed from the Multi-access Edge Computing (MEC) of the used 5G base station.

The main assumptions of the work are the following: (a) an FPGA platform can provide lower latency than more traditionally used GPU platforms for this type of application. This is due to the datapath architecture of the FPGA and DPU, which does not require to first "flood" a large number of Streaming Multiprocessors (SM) as in a GPU; (b) taking advantage of the DPUs, we can reach higher throughputs in terms of the number of processed frames per second; (c) the FPGA platform will provide a better energy efficiency solution compared to CPU and GPU-based solutions.
This paper proposes an edge/cloud telemetry collection and utilization framework for applications where reliability, latency, power efficiency, and high computational capacity are critical. For instance, vehicle safety as well

as vehicular visual and non-visual sensing systems could be potential use cases. We evaluate GPU based platform against the FPGA platform for the role of edge node in an AI computer vision application and set up our framework with the FPGA platform induced by latency and power efficiency numbers provided. We define the cloud-side components of the data lake architecture which will serve later as valuable input for training machine learning networks on the cloud side. In the end, we discuss the reaction time of the cloud side of the framework and FPGA implementation issues which is good to consider when developing AI-based applications on reconfigurable platforms.

Author's contribution: The idea of building a framework for collecting telemetry from multiple sources was discussed and created by all authors. The author suggested the discussion on the type of edge technology to choose in case specific types of applications. The choice of cloud services and experimentation on the cloud and edge parts were performed by the author. Paper writing has been performed by both authors. The presentation of the paper has been done by the author.

5.7 Papers connection

In Figure 5.1, is drawn a diagram which shows the connection between original publications done during this research work.

Publication 1 starts the exploration of possible ways to address the research question. From there derives the implementation part with Publications 2 and 3 where the validation of the model built before is done in different architectures and a real world streaming application. From this flow generates Publication 4 which complements the work done so far by generalizing even more the process of model building. Publication 5 addresses another side of the first question by dealing with overall platform consumption considering not only the MPSoC, while Publication 6 seeks in reaching approaches for achieving better energy-efficiency in edge systems.

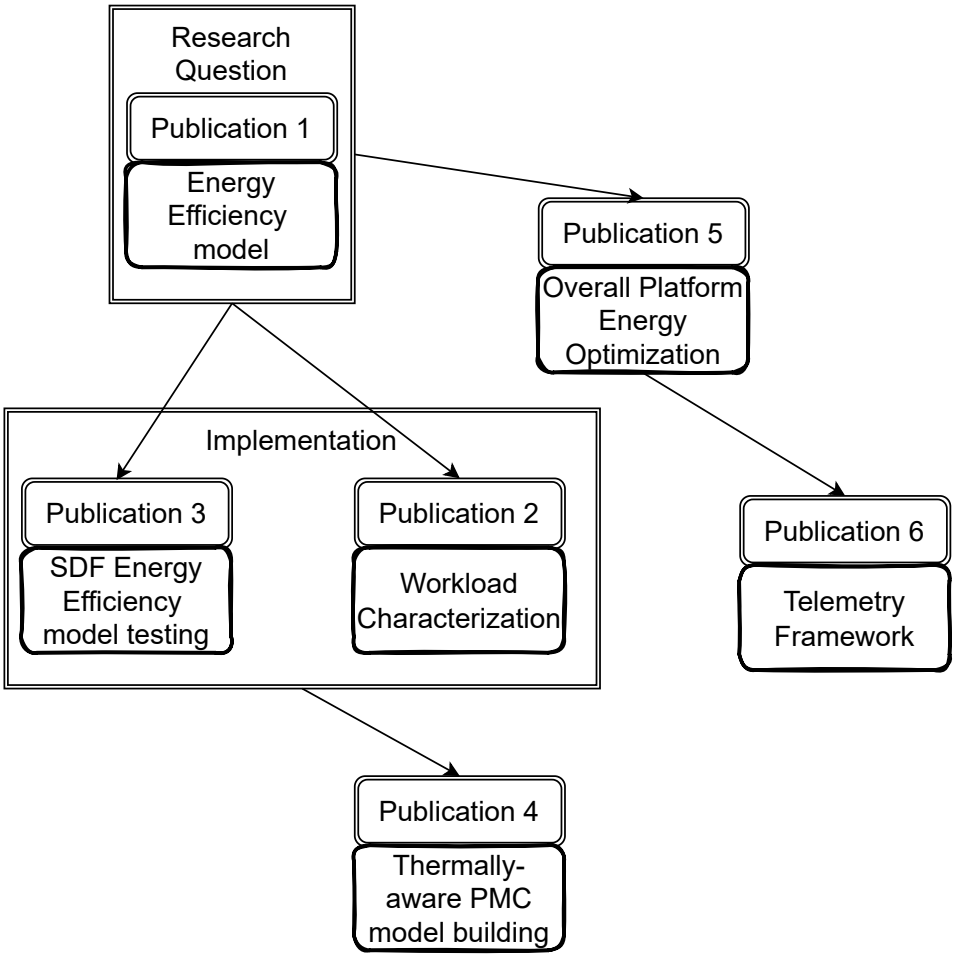

Figure 5.1: Connection between different publications

Chapter 6

Conclusions and Future Work

Throughout the thesis we stressed the need for better energy efficiency in computing systems. Without bringing to much focus on data centers and big server farms, which today account for a small percentage of computing devices meanwhile the number of mobile, and IoT devices, also edge/fog systems is exploding. New applications with new requirements are emerging in every sector like: industry, entertainment, telecom, drone control systems, etc. If in this picture is added the rapid advancement of AI technology due to the high amount of generated data, then the pressure on the execution platforms to be energy efficient is really high. Especially in cases where the mobile platforms are battery powered, or operate on low power levels. Transistor scaling has reached a point where energy efficiency at the transistor level is not providing high benefits as before.

It is clear that the transistor efficiency itself is not enough, and there is a need of cooperation between hardware and software in the challenge to achieve better power proportionality. Mobile platforms today are equipped with powerful MPSoC which recently, as a industry standard, have heterogeneity inside of them as a mechanism to achieve better adaptability between hardware availability and software needs. Knowing the power proportionality problem in computing systems, heterogeneity is a promise to increase the energy spent by doing useful work using the concept of fitness of the software in type of core.

The contribution of this thesis is exploring system actuators and heterogeneity in order to achieve better energy efficiency in mobile multi-core platforms. We introduce the concept of *platform configuration point* to characterize the energy efficiency level of the MPSoC, and also to make optimal choices considering the entire platform energy efficiency, taking into account multiple components of the platform such as memory, disk, networking, etc.

We consider the *utilization rate* of a task as a possible parameter that could provide benefit in the our search for more efficient software execution. A discussion for differences between strategies such as *race-to-idle* and *pace-to-idle* are developed inside the thesis. We discuss the problem of an ever increasing level of heterogeneity implemented in modern multi-core chips, with this resulting in an exponential explosion in the number of platform configuration points that need to be calculated in the energy efficiency model. While also the absence of physical sensors for measuring the power dissipation in the MPSoC, makes it harder to have real-time figures from the power plane. Another contribution of the thesis is providing a methodology for deriving thermally aware energy efficiency models, based on hardware performance counters for estimating the power dissipation of a software task. Including temperature of the chip in the estimation of power, gives better flexibility in making possible software re-configurations.

In today's panorama of the hierarchy of computing systems, there are many steps in which information flows: from sensors/wearables/IoT device, to mobile devices, edge/fog systems, near cloud and far cloud. In making sure to achieve some emerging application requirements like: latency, reliability, adaptability, predictability, and energy efficiency, and also being able to handle the performance requirements of AI applications, edge system have been from some time on the focus of the research and related industries. In order to handle these new requirements, there is the need for the collection of important metrics (telemetry), from different links in the chain flow. In this thesis we propose a telemetry collection and utilization framework, which gives the ability to automatize the work of edge devices while improving some important KPIs like energy efficiency, performance, and platform utilization without service interruption. This framework gives the collaboration of "two worlds", edge and cloud. While the edge is focused in the activity of application execution and telemetry collection, in the cloud happens telemetry monitoring/storage, and the machine learning process in order to improve the AI models that run on the edge.

6.1 Future directions

As an interesting future direction the author foresees the usage of machine learning as the core of the runtime system which collaborates with the system scheduler to correct the application scheduling and also provide the right mapping of the tasks into the execution units of a highly heterogeneous platform. Reinforcement Learning(RL) [78], and Deep Reinforcement Learning(DRL) [79], could be interesting options for both resource management and increasing the energy efficiency of platform units. Designing a strategy with an intelligent agent which learns, the energy efficiency model

of the platform, and the characteristics of the running workloads, while on battery operated devices considering at the same time the SoC(State-of-Charge) could result in a more holistic approach to the energy minimization problem. The objective of the agent will be to suggest the right actuators selection for achieving the optimal energy efficiency, taking into consideration the application performance and also the fitness of the application into the optimal execution engine. Further policy based execution that for example consider extending the battery health or external conditions [80] could be a meaningful addition to the reinforcement learning algorithm.

Bibliography

[1] Simon Holmbacka and Jörg Keller. *Workload Type-Aware Scheduling on big.LITTLE Platforms*, pages 3–17. Springer International Publishing, Cham, 2017.

[2] E. A. Lee and D. G. Messerschmitt. Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245, Sept 1987.

[3] H. Berg, C. Brunelli, and U. Lucking. Analyzing models of computation for software defined radio applications. In *System-on-Chip, 2008. SOC 2008. International Symposium on*, pages 1–4, Nov 2008.

[4] Xiaojun Liu, Yuhong Xiong, and Edward A. Lee. The ptolemy ii framework for visual languages. In *Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)*, HCC '01, pages 50–, Washington, DC, USA, 2001. IEEE Computer Society.

[5] William Thies, Michal Karczmarek, and Saman P. Amarasinghe. Streamit: A language for streaming applications. In *Proceedings of the 11th International Conference on Compiler Construction*, CC '02, pages 179–196, London, UK, UK, 2002. Springer-Verlag.

[6] Soonhoi Ha, Sungchan Kim, Choonseung Lee, Youngmin Yi, Seongnam Kwon, and Young-Pyo Joo. PeaCE: A hardware-software codesign environment for multimedia embedded systems. *ACM Trans. Des. Autom. Electron. Syst.*, 12(3):1–25, August 2007.

[7] Maxime Pelcat, Jonathan Piat, Matthieu Wipliez, Slaheddine Aridhi, and Jean-François Nezan. An open framework for rapid prototyping of signal processing applications. *EURASIP J. Embedded Syst.*, 2009:11:3–11:3, January 2009.

[8] S. Kanur, W. Lund, L. Tsiopoulos, and J. Lilius. Determining a device crossover point in cpu/gpu systems for streaming applications. In

2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 1417–1421, Dec 2015.

[9] Fredric Hällis, Simon Holmbacka, Wictor Lund, Robert Slotte, Sébastien Lafond, and Johan Lilius. Thermal influence on the energy efficiency of workload consolidation in many-core architecture. In Raffaele Bolla, Franco Davoli, Phuoc Tran-Gia, and Tuan Trinh Anh, editors, *Proceedings of the 24th Tyrrhenian International Workshop on Digital Communications*, page 1–6. IEEE, 2013.

[10] Sébastien Lafond. *Simulation of Embedded Systems for Energy Consumption Estimation*. PhD thesis, Turku Centre for Computer Science, 2009.

[11] Simon Holmbacka, Erwan Noguesy, Maxime Pelcaty, Sébastien Lafond, and Johan Lilius. Energy efficiency and performance management of parallel dataflow applications. In *DASIP 2014,Conference on Design & Architectures for Signal & Image Processing*. 2014.

[12] Simon Holmbacka, Sébastien Lafond, and Johan Lilius. Power optimized many-cores with user centric notion of parallelism. In Tomas Nordström and Zain-ul Abdin, editors, *Sixth Swedish Workshop on Multicore Computing*, page 57 – 50. Swedish Multicore Initiative, 2013.

[13] Simon Holmbacka, Dag Ågren, Sébastien Lafond, and Johan Lilius. Qos manager for energy efficient many-core operating systems. In Peter Kilpatrick, Peter Milligan, and Rainer Stotzka, editors, *Proceedings of the 21st International Euromicro Conference on Parallel, Distributed and Network-based Processing*, page 318–322. IEEE Computer society, 2013.

[14] S. Lafond, S. Holmbacka, and J. Lilius. Energy aware software: Issues, approaches and challenges. In *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, pages 1–8, Nov 2016.

[15] Erol Gelenbe and Yves Caseau. The impact of information technology on energy consumption and carbon emissions. *Ubiquity*, 2015(June):1:1–1:15, June 2015.

[16] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[17] Rathijit Sen and David A. Wood. Energy-proportional computing: A new definition. *Computer*, 50(8):26–33, 2017.

[18] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4):23–29, 1999.

[19] Emarketer. 2 billion consumers worldwide to get smart (phones) by 2016. *eMarketer*, 2014.

[20] Minyong Kim, Young Geun Kim, Sung Woo Chung, and Cheol Hong Kim. Measuring variance between smartphone energy consumption and battery life. *Computer*, 47(7):59–65, 2013.

[21] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, 1996.

[22] James Laudon. Performance/watt: The new server focus. *SIGARCH Comput. Archit. News*, 33(4):5–13, November 2005.

[23] Juan-Antonio Carballo, Wei-Ting Jonas Chan, Paolo A Gargini, Andrew B Kahng, and Siddhartha Nath. Itrs 2.0: Toward a re-framing of the semiconductor technology roadmap. In *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pages 139–146. IEEE, 2014.

[24] Etienne Sicard and Lionel Trojman. Introducing 5-nm FinFET technology in Microwind, June 2021. This paper describes the implementation of a high performance FinFET-based 5-nm CMOS technology in Microwind. After a general presentation of the electronic market and the roadmap to 1nm technology, design rules and basic metrics for the 5-nm node are presented. Concepts related to the design of FinFET and design for manufacturing are also described. The performances of a ring oscillator, basic cells and a 6-transistor RAM memory are also analyzed.

[25] J Moyne. International technology roadmap for semiconductors (itrs) factory integration, 2015: Summary of updates and deep dive into big data enhancements. In *Proceedings of the APC Conference XXVII, Austin, TX, USA*, pages 12–15, 2015.

[26] Anantha Chandrakasan. Design of high-performance microprocessor circuits. *IEEE*, 2000.

[27] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama. Variable supply-voltage scheme for low-power high-speed cmos digital design. *IEEE Journal of Solid-State Circuits*, 33(3):454–462, 1998.

[28] Ruchir Puri. Minimizing power under performance constraint. In *2004 International Conference on Integrated Circuit Design and Technology (IEEE Cat. No. 04EX866)*, pages 159–163. IEEE, 2004.

[29] Pushpa Saini and Rajesh Mehra. Leakage power reduction in cmos vlsi circuits. *International Journal of Computer Applications*, 55(8), 2012.

[30] 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)*, pages 5 pp.–637, 2006.

[31] Anton Shilov. Tsmc outlines 2nm plans: N2p brings backside power delivery in 2026, n2x added to roadmap, Apr 2023.

[32] Changwoo Noh, Changwoo Han, Sang Min Won, and Changhwan Shin. Vertical gate-all-around device architecture to improve the device performance for sub-5-nm technology. *Micromachines*, 13(9), 2022.

[33] Samuel K. Moore. Meet the forksheet: Imec's in-between transistor, May 2023.

[34] Hergys Rexha and Sébastien Lafond. Exploring energy efficiency model generalization on multicore embedded platforms. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 494–498, 2018.

[35] N. Pinckney, R. G. Dreslinski, K. Sewell, D. Fick, T. Mudge, D. Sylvester, and D. Blaauw. Limits of parallelism and boosting in dim silicon. *IEEE Micro*, 33(05):30–37, sep 2013.

[36] Peter Greenhalgh. Big. little processing with arm cortex-a15 & cortex-a7: Improving energy efficiency in high-performance mobile platforms. *white paper, ARM Ltd*, 2011.

[37] Hsinchen Chen, Rolf Lagerquist, Ashish Nayak, Hugh Mair, Gokulakrishnan Manoharan, Ericbill Wang, Gordon Gammie, Efron Ho, Anand Rajagopalan, Lee-Kee Yong, Ramu Madhavaram, Madhur Jagota, Chi-Jui Chung, Sudhakar Maruthi, Jenny Wiedemeier, Tao Chen, Henry Hsieh, Daniel Dia, Amjad Sikiligiri, Manzur Rahman, Barry Chen, Curtis Lin, Vincent Lin, Elly Chiang, Cheng-Yuh Wu, Po-Yang Hsu, Jason Tsai, Wade Wu, Achuta Thippana, and SA Huang. 4.1 a 7nm 5g mobile soc featuring a 3.0ghz tri-gear application processor subsystem. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 64, pages 54–56, 2021.

[38] S Keckler. Life after dennard and how i learned to love the picojoule. *Keynote at MICRO*, 2011.

[39] G. Semeraro, D.H. Albonesi, S.G. Dropsho, G. Magklis, S. Dwarkadas, and M.L. Scott. Dynamic frequency and voltage control for a multiple clock domain microarchitecture. In *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. (MICRO-35). Proceedings.*, pages 356–367, 2002.

[40] Sebastian Herbert and Diana Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the 2007 International Symposium on Low Power Electronics and Design*, ISLPED '07, page 38–43, New York, NY, USA, 2007. Association for Computing Machinery.

[41] Stefanos Kaxiras and Margaret Martonosi. 2008.

[42] Aaron Carroll and Gernot Heiser. Unifying dvfs and offlining in mobile multicores. In *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 287–296, 2014.

[43] Aaron Carroll and Gernot Heiser. Mobile multicores: Use them or waste them. *SIGOPS Oper. Syst. Rev.*, 48(1):44–48, may 2014.

[44] Kishore Kumar Pusukuri, Rajiv Gupta, and Laxmi N. Bhuyan. Thread reinforcer: Dynamically determining number of threads via os level monitoring. In *2011 IEEE International Symposium on Workload Characterization (IISWC)*, pages 116–125, 2011.

[45] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities, reprinted from the afips conference proceedings, vol. 30 (atlantic city, n.j., apr. 18–20), afips press, reston, va., 1967, pp. 483–485, when dr. amdahl was at international business machines corporation, sunnyvale, california. *IEEE Solid-State Circuits Society Newsletter*, 12(3):19–20, 2007.

[46] Wonyoung Kim, Meeta S. Gupta, Gu-Yeon Wei, and David Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134, 2008.

[47] Mohammed A. Noaman Al-hayanni, Ashur Rafiev, Fei Xia, Rishad Shafik, Alexander Romanovsky, and Alex Yakovlev. Parma: Parallelization-aware run-time management for energy-efficient many-core systems. *IEEE Transactions on Computers*, 69(10):1507–1518, 2020.

[48] Jonathan Corbet. Per-entity load tracking. *LWN article, available at: https://lwn. net/Articles/531853*, 2013.

[49] Neil Audsley, Alan Burns, Rob Davis, Ken Tindell, and Andy Wellings. Real-time system scheduling. In Brian Randell, Jean-Claude Laprie, Hermann Kopetz, and Bev Littlewood, editors, *Predictably Dependable Computing Systems*, pages 41–52, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[50] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, jan 1973.

[51] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*, pages 4–13, 1998.

[52] cgroups - linux control groups.

[53] cpuacct.

[54] Arpan Gujarati, Felipe Cerqueira, and Björn B. Brandenburg. Outstanding paper award: Schedulability analysis of the linux push and pull scheduler with arbitrary processor affinities. In *2013 25th Euromicro Conference on Real-Time Systems*, pages 69–79, 2013.

[55] Simon Holmbacka, Fredric Hällis, Wictor Lund, Sébastien Lafond, and Johan Lilius. Energy and power management, measurement and analysis for multi-core processors. Technical Report 1117, 2014.

[56] O. J. Arndt, D. Becker, C. Banz, and H. Blume. Parallel implementation of real-time semi-global matching on embedded multi-core architectures. In *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 56–63, July 2013.

[57] D. Liu, J. Spasic, G. Chen, and T. Stefanov. Energy-efficient mapping of real-time streaming applications on cluster heterogeneous MPSoCs. In *2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, pages 1–10, October 2015.

[58] Kenzo Van Craeynest, Aamer Jaleel, Lieven Eeckhout, Paolo Narvaez, and Joel Emer. Scheduling Heterogeneous Multi-cores Through Performance Impact Estimation (PIE). In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, pages 213–224, Washington, DC, USA, 2012. IEEE Computer Society.

[59] Santanu Sarma, T. Muck, Luis A. D. Bathen, N. Dutt, and A. Nicolau. SmartBalance: A Sensing-driven Linux Load Balancer for Energy Efficiency of Heterogeneous MPSoCs. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 109:1–109:6, New York, NY, USA, 2015. ACM.

[60] H. Mair, E. Wang, A. Wang, P. Kao, Y. Tsai, S. Gururajarao, R. Lagerquist, J. Son, G. Gammie, G. Lin, A. Thippana, K. Li, M. Rahman, W. Kuo, D. Yen, Y. C. Zhuang, U. Fu, H. W. Wang, M. Peng, C. Y. Wu, T. Dosluoglu, A. Gelman, D. Dia, G. Gurumurthy, T. Hsieh, W. Lin, R. Tzeng, J. Wu, C. Wang, and U. Ko. 3.4 A 10nm FinFET 2.8ghz tri-gear deca-core CPU complex with optimized power-delivery network for mobile SoC performance. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 56–57, February 2017.

[61] Young Duk Kim, Wookyeong Jeong, Lakkyung Jung, Dongsuk Shin, Jae Geun Song, Jinook Song, Hyeokman Kwon, Jaeyoung Lee, Jaesu Jung, Myungjin Kang, Jaehun Jeong, Yoonjoo Kwon, and Nak Hee Seong. 2.4 a 7nm high-performance and energy-efficient mobile application processor with tri-cluster cpus and a sparsity-aware npu. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 48–50, 2020.

[62] ARM big.LITTLE white paper.

[63] Hiroshi Sasaki, Satoshi Imamura, and Koji Inoue. Coordinated power-performance optimization in manycores. In *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, pages 51–61, 2013.

[64] Simon Holmbacka. Energy aware software for many-core systems. 2015.

[65] Connor Imes and Henry Hoffmann. Minimizing Energy Under Performance Constraints on Embedded Platforms: Resource Allocation Heuristics for Homogeneous and single-ISA Heterogeneous Multi-cores. *SIGBED Rev.*, 11(4):49–54, January 2015.

[66] Energy Efficient Scheduling of Real Time Signal Processing Applications through Combined DVFS and DPM. pages 622–626, February 2016.

[67] Hongsuk Chung, Munsik Kang, and Hyun-Duk Cho. Heterogeneous multi-processing solution of exynos 5 octa with arm big. little technology. *Samsung White Paper*, 2012.

[68] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete. Cycle-static dataflow. *Trans. Sig. Proc.*, 44(2):397–408, February 1996.

[69] Manfred Broy. A theory for nondeterminism, parallelism, communication, and concurrency. *Theoretical Computer Science*, 45:1 – 61, 1986.

[70] J. S. Lee, K. Skadron, and S. W. Chung. Predictive Temperature-Aware DVFS. *IEEE Transactions on Computers*, 59(1):127–133, January 2010.

[71] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett. Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(1):106–119, January 2017.

[72] gSM Association et al. The mobile economy 2020. *GSMA HEAD OFFICE*, 2020.

[73] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[74] Yunbo Li, Anne-Cécile Orgerie, Ivan Rodero, Betsegaw Lemma Amersho, Manish Parashar, and Jean-Marc Menaud. End-to-end energy models for Edge Cloud-based IoT platforms: Application to data stream analysis in IoT. *Future Generation Computer Systems*, 87:667–678, October 2018.

[75] E. Rotem, U. C. Weiser, A. Mendelson, R. Ginosar, E. Weissmann, and Y. Aizik. H-EARtH: Heterogeneous Multicore Platform Energy Management. *Computer*, 49(10):47–55, October 2016.

[76] Henry Hoffmann. Racing and pacing to idle: An evaluation of heuristics for energy-aware resource allocation. In *Proceedings of the Workshop on Power-Aware Computing and Systems*, HotPower '13, New York, NY, USA, 2013. Association for Computing Machinery.

[77] Xilinx. Xilinx powers alibaba cloud faas with ai acceleration solution for e-commerce business.

[78] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[79] Hongjia Li, Tianshu Wei, Ao Ren, Qi Zhu, and Yanzhi Wang. Deep reinforcement learning: Framework, applications, and embedded implementations: Invited paper. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 847–854, 2017.

[80] Takao Moriyama, Giovanni De Magistris, Michiaki Tatsubori, Tu-Hoa Pham, Asim Munawar, and Ryuki Tachibana. Reinforcement learning testbed for power-consumption optimization. In *Methods and Applications for Modeling and Simulation of Complex Systems: 18th Asia Simulation Conference, AsiaSim 2018, Kyoto, Japan, October 27–29, 2018, Proceedings 18*, pages 45–59. Springer, 2018.

Part II

ORIGINAL PUBLICATIONS

Paper 1

Core Level Utilization for Achieving Energy Efficiency in Heterogeneous Systems

1

Hergys Rexha, Simon Holmbacka, Sébastien Lafond

Originally published *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 401-407*

Core Level Utilization for Achieving Energy Efficiency in Heterogeneous Systems

*Hergys Rexha, Simon Holmbacka †, Sébastien Lafond‡
Faculty of Science and Engineering, Åbo Akademi University, Turku Centre for Computer Science, Turku, Finland
Email: *hergys.rexha@abo.fi, †simon.holmbacka@abo.fi, ‡sebastien.lafond@abo.fi

***Abstract*—Energy budget is becoming a constraint in all computing systems. From mobile systems to supercomputers, the focus has shifted from performance to energy and power efficiency. Design metrics are not anymore solely based on performance, as the energy efficiency of application executions is becoming a predominant design requirement. In addition to established voltage and frequency scaling techniques, several semiconductor chip manufactures introduced heterogeneous multi-core processors to increase the level of energy efficiency. The usage of this heterogeneity is complicated by the scheduling and mapping decisions needed to be made at run-time for application execution. In order to exploit the full potential of such architectures we need to make the right decisions, because parameters such as type of core, frequency and utilization usually affect the power dissipation and performance. This paper analyses achievable energy gains when exploiting core level utilization in addition to other control techniques such as: heterogeneity, voltage and frequency scaling. We build an energy efficiency model based on platform configurations defined by core types, the different voltage and frequency levels and the core utilization rate. Based on the built model, we analyze the energy efficiency variations for different platform configurations providing the same level of performance. We show that trading the number and type of core with frequency and voltage level and core utilization rate can lead to substantial energy efficiency gains.**

I. Introduction

According to the advisory company Gartner, Inc the handheld device market is one of the fastest growing sectors in the computer industry. Battery operated devices are continuously facing different type of workloads with different performance requirements. All these variances in software should be executed on hardware with sometimes a limited energy budget. The limited amount of energy is one reason why many of such computing devices are taking advantages of multiple processors system on chip (MPSoC) [1]. The design space of such platforms is very broad [2] and explored by researchers with the conclusion that multi-core heterogeneity can provide energy savings [3]. Mostly savings come from the appropriate choice of execution unit to use, for a specific job.

The next rising problem is how to map the execution of parallel applications on multi-core platforms. This has been an investigated area with a rich variety of objectives [4] [1] . Some of the key objectives are: performance, fairness, predictability, reliability etc...

One of the strategic objectives considered with the use of MPSoC is energy. Energy consumption is an intrinsic metric that is dependent on multiple factors. Beside the ratio of dynamic and static power dissipation, the execution time of the application and the physical architectural features must be considered in order to define a winning strategy for minimizing energy consumption. As the type of workload is diverse, the optimal choice for mapping software to hardware becomes a challenging problem. One of the latest techniques explored by the research community involves using heterogeneity in order to achieve better performance and energy levels. Because the consideration of parameters such as type of core, frequency and core utilization rate affect the power dissipation, in this paper we are going to explore the impact of these arguments on the energy efficiency of heterogeneous MPSoC. More precisely we investigate how, dynamic voltage and frequency scaling (DVFS), core heterogeneity and core utilization influence the overall energy efficiency of heterogeneous MPSoC based computing systems.

II. Scope of the work

With the improvements provided in the semiconductor production technology, industry is able to provide an impressive level of integration at transistor level. This process gave birth to MPSoC. However with post Dennard scaling [5] the power density increased with every processor generation. Recent studies show that after 2005 for the same chip area the power dissipation increased by a factor of 2 [6] for every process technology. That is why in the recent years, industry has turned towards architectures with multiple core types on a single chip. These architectures can offer different levels of programmable logic beside conventional cores, which altogether can result in a more convenient choice compared to symmetric architectures [7, 8]. The platforms based on this technology are able to offer execution to a wide range of workloads, varying from memory databases (requiring a small computation power) to multimedia applications which can be computational hungry. Mapping workloads on many-core architectures has followed different objectives such as: performance, latency, throughput and reliability [9, 4].
One of the latest concerns is mapping for energy efficiency. In addition to power gating unused cores, the conventional techniques for achieving energy conservation are DVFS and the use of different core types in modern architectures, which are called heterogeneous microarchitectures (HM) [3].

A typical heterogeneous architecture is the big.LITTLE architecture introduced by ARM [10]. This architecture is equipped with two type of cores, one optimized for performance and the other optimized for energy efficiency. The cores are organized as a MPSoC and have different microarchitectural organizations but share the same ISA, which provides a binary compatibility between the two types of cores. In addition to the power reduction offered by DVFS and core power gating, exploiting the distinct characteristics of the different cores makes the big.LITTLE architecture an attractive platform to achieve high levels of energy efficiency.

As Imes and Hoffmann conclude in their study [11] for achieving energy efficiency in heterogeneous architecture under a moderate load, load levels on core should be kept constant. In other words we should try to keep the cores busy for most of the time. A strategy coined with the name "never-to idle". Using the never-to-idle strategy scheduling heuristic, we search for the near optimal energy efficient configuration between the number, type and frequency level of cores, considering as a requirement the performance characteristic of the tasks to map. In [12] authors build an energy model based on real hardware measurements on HMP and use a convex optimization framework for decide about the efficient trade off between DVFS and DPM. As a validation mechanism they use a dataflow signal processing application for their predictability. In difference we propose including the utilization level for making the right decision scheduling. We consider a system

with full HMP instead of a cluster scheduling considered from the authors. This gives us more flexibility in the scheduling decisions.

This paper considers the possibility to control the load level of each core to improve further the energy efficiency under different performance constrains. As a performance constraint we consider levels of throughput required. Recently new Linux scheduling framework, called sched_deadline, provides the possibility to define the utilization level of tasks. The sched_deadline scheduler is now available in most of the standard Linux distributions. As a new element in our study we will explore task utilization parameter to obtain a near optimal configuration. In this paper we consider a clustered MPSoC composed of two independent clusters, one optimized for performance and the other for energy, where each cluster can have a single voltage and corresponding frequency level. A schematic of the heterogeneous architecture is shown in Figure 1.

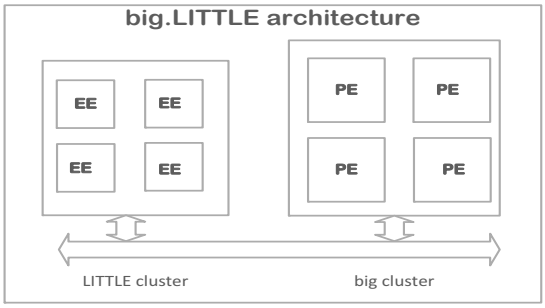

Fig. 1: Schematic of heterogeneous architecture

In this paper we answer the question *which platform configuration provides the most energy efficient execution under different performance constraints?* under the following assumptions:

- the application has a configurable level of parallelism
- the hardware platform that can activate different type of core organized in clusters
- the platform provides as actuators different DVFS levels and core utilization rates

In this work the core utilization rate is also considered as a parameter indeed, taking advantage of the recently introduced sched_deadline scheduler [13]. Indeed, previous work already demonstrated that core utilization rate can have a clear impact on the power efficiency of multi-core architectures [14].

We define the platform configuration as the:

- number of parallel instances from the application to be executed
- number and type of cores to utilize
- DVFS level of the clusters
- load level (utilization rate) of each cluster.

The configuration analysis will be based on specific performance levels required by the application. Taking into account the number of DVFS level for each core, the number of cores and utilization levels each cluster is able to provide, there is a large number of possible configurations to consider. For example, if we have N cores divided in two types of cluster N_b and N_L, each core type has F_b and F_L frequency levels and the levels of utilization of the tasks are L. Then the number of configurations will be:

$$C = N_b * N_L * F_b * F_L * L_b * L_L + N_b * F_b * L_b + N_L * F_L * L_L$$

In this work we consider that DVFS can be applied at the cluster level and that all tasks mapped to a cluster have the same level of utilization (load level). Taking into consideration the former constraints the cardinality of the configuration space is |C|=64 008.

III. Related work

There is a large body of work focusing on energy efficiency. The closest to our work is the paper from Sozzo and Durelli [15]. In their work the authors suggest a runtime system which monitors the throughput of the applications through heartbeat framework with the goal of minimizing the power under a performance requirement. They consider in the configuration space, only the type of core to use and the frequency level. Differently here we also consider the level of utilization as an additional parameter in the configuration space. Moreover, they proposed an analytic model for calculating the power dissipation, while in our work we build power model based on experimental data. In [16], authors discuss about parallelism inside the application and how it can be exploited in order to achieve better energy efficiency. Parallel sections of the application are not known during execution, as in some case even the minimum level of required performance is unknown. By expressing explicitly these two parameters in the application source code, in various phases, a run-time power manager can make the optimal decisions for resource allocation. In contrast with this work, in our paper beside DVFS and the level of parallelism, we exploit heterogeneity, which give the possibility to choose between different core types, to evaluate the possibility to achieve better energy efficiency.

In [17] authors present a scheduling and mapping algorithm that decides between heterogeneity and DVFS for executing streaming applications in clustered MPSoC. The examples mentioned are modeled as Dataflow applications which need to be executed above a certain level of throughput and latency. The algorithm is able to adapt to the changes in execution by removing tasks from the least used cluster in order to shut them down. In contrast with this work we do not limit our focus to only hard real time applications and explore the parallelism inside the application in order to make an optimal mapping decision. We also take into account the level of utilization of the core, in selecting an optimal configuration. Lukefahr and Padmanabha [3], consider the effectiveness of two techniques for achieving energy efficiency. DVFS and heterogeneous michroarchitectures are the focus of the study where they try to solve the following problem: at which granularity during software execution should the decisions be made about the use of former mentioned techniques? They propose an analysis tool called PaTH which defines an optimal schedule for achieving Pareto-optimal energy savings, for a given architecture. At the end they come to the conclusion that a combination of the two techniques is better for achieving energy efficiency, in case of coarse granularity in software phases. Differently from this work we explore the impact of core utilization on the scheduling and mapping decisions for achieving a set of performance levels.

The fitness of a workload into different core types inside a heterogeneous system, is explored by Van Craeynest and Jaleel in [1]. They propose a mechanism to try to find the best mapping between workload and core type in order to get an efficient execution. By collecting metrics from workload profiling information they try to estimate the performance impact when the task is mapped on a specific type of core. The results show better scheduling decisions which will affect the performance and energy consumption. Differently in our paper we do not use any performance counters for estimating the power dissipation but base our approach on power models constructed from experimental data. In the work of Santanu and Muck [18] are considered different heterogeneous architectures. They propose a control system in order to overcome the problems encountered by standard Linux scheduler which is unable to deal with high degree of heterogeneity.

The approach is composed by three phases called: sensing, estimation, and balance. The balancing mechanism is executed for each scheduling epoch and manages efficiently the architectural resources following dynamic changes in the workload. In our work we focus on architectures that are composed of two type of cores that can be enabled simultaneously in a more lightweight process.

The usage of DVFS and thread scheduling has been well studied in different works. In [19] authors try to improve the power efficiency through frequency scaling and scheduling. They propose a technique for estimating the power efficiency metric in different application phases, with various frequencies and core types. This technique uses hardware performance counters (HPC) like number of fetched and retired instructions, cache hits/misses, number of predicted branches and IPC. Instead of focusing on performance counters for power efficiency we concentrate on energy efficiency as a more valuable metric (especially for battery operating devices).

Price theory economics model has been introduced in [20] to make the right decision for achieving a performance level under the minimum power requirement. The proposed framework is distributed and coordinates DVFS, task migration and load balancing for achieving the specified performance under a certain thermal design (TPD). The framework is composed by different agents (core agent, cluster agent, task agent and chip agent) which interact with each other to define the right price for the computation unit cycles.

In contrast with the previous works, our work focuses on energy efficiency while providing different application level performances. We consider the impact of utilization level variances in energy efficiency when selecting platform configurations.

IV. Experimental settings

In our experiments we use the ODROID XU4 development board from HARDKERNEL which is powered by a Samsung Exynos 5422 octa core SoC. The CPU is organized with two clusters, one with 4 ARM Cortex A7 and the other with 4 ARM Cortex A15. Clusters work independently using HMP (Heterogeneous Multi Processing). Cores share the same instruction set architecture (ISA) but have different michroarchitecture, giving A7 cores a more lightweight implementation of the pipeline. For measuring the power dissipated by the ODROID we used a RaspberryPI connected through its I2C pins to the power supply of the ODROID board. On the RaspberryPi we run a Datalogger software which samples current and voltage values every 100 ms. We use similar measuring infrastructure as in [21]. The ODROID development board is installed with Linux version 4.2.0-rc7-74, gcc version 4.9.2. The board is connected through an Ethernet port without any other connection, in order to minimize the power dissipated by the board. For achieving more accuracy from measurements we removed the power dissipated by the active cooling of the board. The system will run only the essential services.

In order build our models and follow the strategy of not executing with maximum load to the highest frequency, as suggested by experiments in [16], we need a workload generator able to maintain a constant utilization level on the cores. To build our power and energy models, we use Spurg-bench [21] which is a multi-core load generator written in python, able to sustain various utilization levels in terms of operations per second on different number of threads. Spurg-bench supports as parameters the number of operations to be executed, the rate of execution to be sustained and the number of instances (threads) to achieve that utilization. We use floating point operations as the work load generated by Spurg-bench.

For each experiment we run the workload with utilization levels from 10% to 90%, avoiding the run at 100%. We increase by 10% the utilization every time. First, with one instance of Spurg-bench running on the A7 cluster having only one core, we explore the entire range of available frequencies with a varying utilization from 10% to 90%. Then a similar run is performed on the A15 cluster. For each experiment we log the power data with the RaspberryPI, and collect also the performance data from the application in terms of operations per second. Each data point in the graphs is obtained by running experiments 10 times. For each frequency and each utilization level we take the average power dissipation from the logs. In the second stage of the experiments we measure the power dissipated by running multiple threads in each cluster. A work flow schematic of the experiment is shown in Figure 2. The frequency governor *cpufreq* in Linux gives the possibility

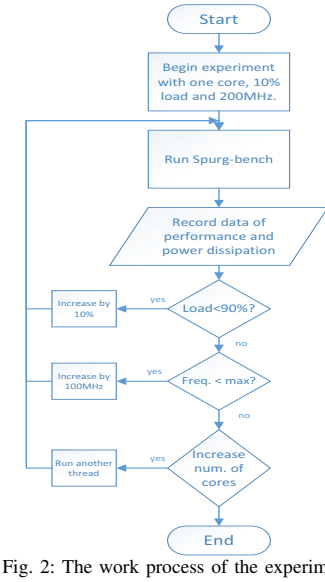

Fig. 2: The work process of the experiment

to define 14 frequency levels on the A7 cores, from 200MHz to 1.4GHz and 18 levels on the A15 cores from 200MHz to 2GHz. With these intervals correspond 4 discrete voltage levels for driving the cores. The voltage and frequency levels for both cores are shown in Tables II and I.

Core frequency (MHz)	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400	1500	1600	1700	1800	1900	2000
Core voltage	0.9V							1V				1.1V			1.25V				

TABLE I: Frequency and voltage relation for A15

Core frequency (MHz)	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400
Core voltage	0.9V		1V				1.1V			1.25V			

TABLE II: Frequency and voltage relation for A7

A diagram of the components involved in the experiments can be seen on the Figure 3.

V. Models

We built a power model for each core type using experimental settings described in the previous section. Based on

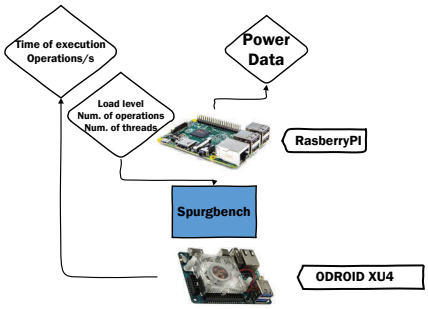

Fig. 3: Experimentation infrastructure

the obtained power model, we derive the energy efficiency of the different cores at different utilization (load) levels. We then list in a table the energy efficiency and corresponding performance level for all possible platform configurations. We can therefore use this table to select the most energy efficient configurations for level of performance.

A. Power models

In modeling power dissipation for the big.LITTLE architecture, we did choose a top down approach based on measurements done directly on the hardware. The advantages of this approach compared to bottom up are the accuracy level and reliability. For each type of core, a model is derived under different frequencies and utilization levels. As mentioned previously these models are obtained using as a workload Spurgbench with floating point operations. Floating point operations generally exercise most of the pipeline michroarchitecture. To build the models, we first measure the power dissipated by both clusters with different number of cores engaged at different levels of load. The results for different number of cores are shown in Figures 4 and 5. Because of space constraint we choose to show only one frequency for each cluster.

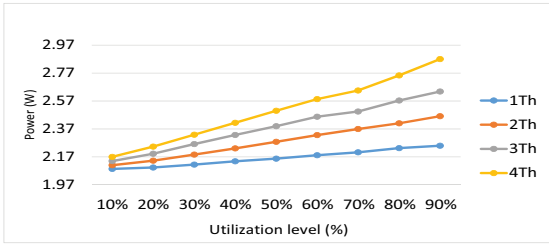

Fig. 4: Power dissipation for LITTLE cluster at 1.3 GHz

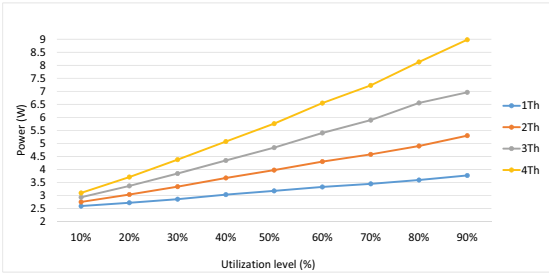

Fig. 5: Power dissipation for big cluster at 1.9 GHz

As described on Figure 2, to obtained the core level power dissipation we sequential increase the number of cores. The core level power dissipation is derived by subtracting the different measured cluster-level power dissipations. The results of the power dissipated by the two different cores under different levels of utilization are shown in Figures 6 and 7. From the figures we can observe for different frequency (and corresponding voltage) the near-linear relationship between utilization levels and power dissipation.

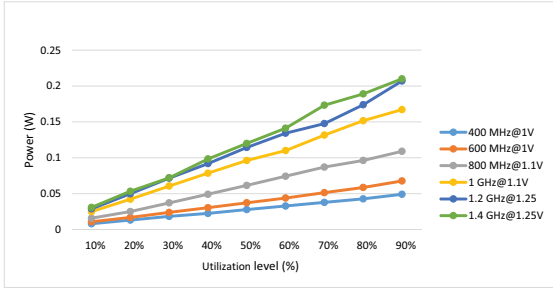

Fig. 6: Power dissipation at the core level of ARM Cortex-A7

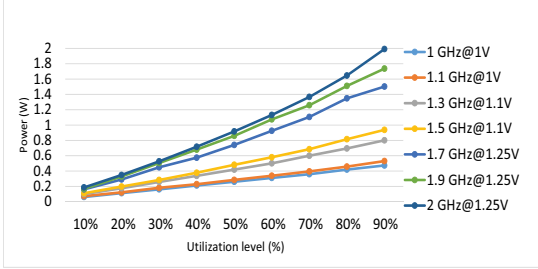

Fig. 7: Power dissipation at the core level of ARM Cortex-A15

B. Energy efficiency results

From the power measurement data we can asses the energy efficiency of each core type under different levels of utilization and different frequencies. We express the energy efficiency using the achieved number of operations per joule metric. For different utilization and frequency levels, we derive the corresponding core level energy efficiency. Figure 8 and 9 show the energy efficiency of one A7 and A15 core respectively. From the figures we observe for all frequency a non-linearity of the energy efficiency. Therefore, we can expect different efficiency level for different platform configurations if the utilization rate of the core can be controlled. From the figures we notice the efficiency variation range from around 10% to 60% depending on the frequencies and type of core.

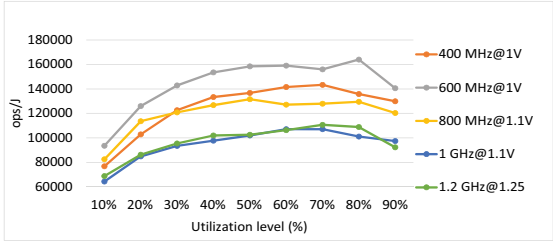

Fig. 8: Energy efficiency of ARM Cortex-A7 core

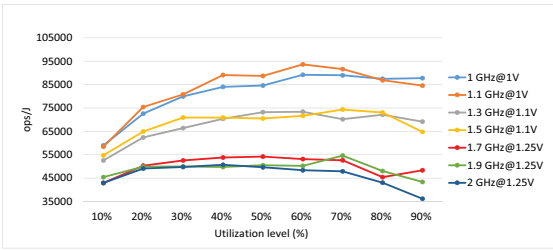

Fig. 9: Energy efficiency of ARM Cortex-A15 core

C. Energy efficiency tables

From the measurements and derived efficiency values, we can construct what we call an energy efficiency table as shown in Table III . The table has three columns. The first specifies the type of configuration we have in terms of active processing unit type, utilization sustained and frequency level. The second refers to the performance level of the corresponding configuration. The third column specifies the dissipated power for that configuration. The fourth column contains the energy efficiency value derived from the two previous columns. The efficiency table includes the full configuration space, listing all possible combinations between number of cores in each cluster, core type, frequency level and core utilization.

TABLE III: Energy efficiency table .

C(L_b/F_b/N_b/L_l/F_l/N_l)	Perf.(op/s)	P(W)	Efficiency(op/J)
90%/2 GHz/4A15/90%/1.4 GHz/4A7	377 886	8.80	572 154
90%/1.9 GHz/4A15/90%/1.2 GHz/4A7	377 401	7.77	542 614
90%/2 GHz/4A15/80%/1.4 GHz/4A7	376 122	8.72	610 402
90%/1.9 GHz/4A15/90%/1.4 GHz/4A7	390 718	7.78	600 850
90%/1.7 GHz/4A15/90%/1.4 GHz/4A7	380 526	6.84	621 047
80%/1.9 GHz/4A15/90%/1.4 GHz/4A7	379 898	6.88	619 590
90%/1.9 GHz/4A15/70%/1.4 GHz/4A7	379 015	7.63	623 948
90%/1.7 GHz/4A15/80%/1.2 GHz/4A7	366 459	6.70	629 088
70%/1.9 GHz/4A15/90%/1.4 GHz/4A7	364 930	5.87	646 022
90%/1.9 GHz/4A15/30%/1.4 GHz/3A7	325 356	7.15	512 412
80%/2 GHz/4A15/50%/1.4 GHz/3A7	325 036	6.94	518 683
.	.	.	.

Therefore, we sort the table according to the efficiency level of all configurations. A view of the resulting table, sorted in descending order, is given in Table IV. Because of the lack of space we show only the beginning and the end of the table.

TABLE IV: Ordered Energy efficiency table .

C(L_b/F_b/N_b/L_l/F_l/N_l)	Perf.(op/s)	P(W)	Efficiency(op/J)
60%/1.1 GHz/4A15/60%/200 MHz/4A7	136 790	1.39	1 287 932
60%/1.1 GHz/4A15/80%/200 MHz/4A7	140 511	1.41	1 285 211
60%/1.1 GHz/4A15/70%/200 MHz/4A7	138 377	1.40	1 283 870
70%/1.1 GHz/4A15/60%/200 MHz/4A7	154 722	1.62	1 279 862
70%/1.1 GHz/4A15/80%/200 MHz/4A7	158 443	1.64	1 277 141
.	.	.	.
10%/2 GHz/1A15	8 000	0.18	42 962
10%/1.7 GHz/1A15	6 922	0.16	42 890
90%/2 GHz/1A15	72 028	1.99	36 165

Using the obtained sorted table, we can look up though the configuration space to find the most energy efficient solution for a required application performance. The selected configuration will define the level of parallelism, the number and type of cores and the corresponding level of utilization to achieve the most energy efficient execution at the requested performance level.

In Figure 10 we show the power dissipation of ten performance groups which are composed of various configuration options. For illustration purpose, the presented performance groups are only a subset of those obtainable from the table.

For the same ten performance groups, Figure 13 shows the corresponding energy efficiency results. On the figures each performance group is composed by different configuration solution providing the same performance level (expresses in operations per second).

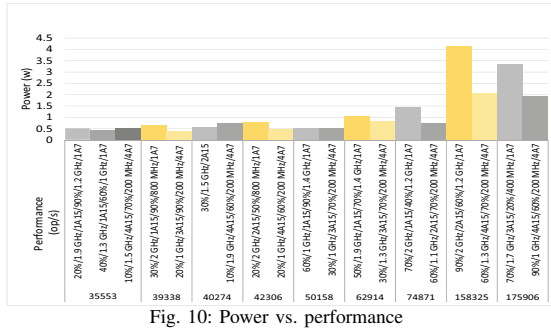

Fig. 10: Power vs. performance

We observe that depending on the selected configurations the variation in energy efficiency can be considerable. For example when requesting a performance level of 74871 operations per second, the difference in energy efficiency between possible configurations is reaching 631 %. For this particular performance level, we observe that the increase in efficiency is provided by adding one A15 and three A7 cores, having the cores clocked at a lower frequency, decreasing the utilization level on the A15 and increasing it on the A7. In other words, the number of cores was traded for lower clock frequencies and different utilization levels.

From the energy model we can plot the efficiency levels as a function of performance for all possible configurations. Figure 11 shows the best and worst efficiency configurations for all reachable performance levels. From the Figure we can observe that up to the performance level of $2e^{+}5$ operations per second the density of configuration points is high and the difference between the efficiency of the worst and best configurations is large. Moreover, up to this performance level there are configurations providing the highest possible efficiency level. After this performance level the configuration density is decreasing and the highest possible efficiency levels are not reachable anymore.

VI. Application Case Study

In the previous section we presented our approach to built the energy efficiency model and table, using the synthetic load generator Spurg-bench. Using the proposed approach and exploiting the newly sched_deadline policy, we show in this section achievable results for an application case study.

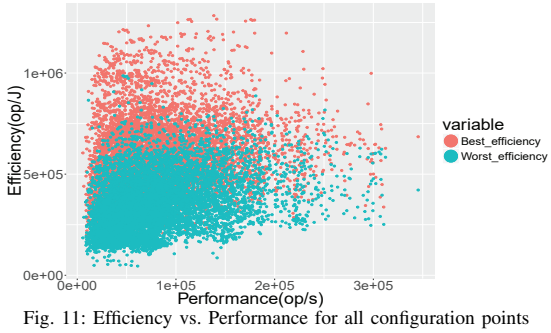

Fig. 11: Efficiency vs. Performance for all configuration points

A. Sched_deadline algorithm

In order to implement our strategy taking into account the utilization rate of a task on a specific core, we need to assign to target tasks a specific level of utilization which has to be sustained during execution. There is no direct way in common Linux schedulers to enforce such a constraint. Lately a different scheduler framework has been introduced in mainstream Linux. It offers the possibility to explicitly express the level of utilization of a task. For each task the sched_deadline policy needs three parameters: the task runtime, its period and deadline. Therefore, by assigning a specific task runtime value we can obtain a corresponding task utilization level.

B. Model validation

For evaluating our approach we consider an application which can express different levels of parallelism and have specific deadlines. We use the Blackscholes applicaton from the PARSEC benchmark as a case study. This application calculates the prices of an European options portfolio, based on a partial differential formula. The Blackscholes benchmark represents a wide range of partial derivative equation (PDE) solvers which are used in financial analysis [22]. The application takes as an input a portfolio with a number of options. Then the portfolio, divided between threads, is processed concurrently. The application contains mainly floating point operations and is CPU intensive. The application is largely scalable and has a small parallelization overhead. The application executes as the following: at the beginning the main thread initializes the portfolio of options, then it spawns worker threads which process part of the data in a parallel. Each thread executes *BlkSchlsEqEuroNoDiv* function to compute options prices. During the experiments we use up to 8 threads in the parallel region of the application and only measure the behavior in that region, which is called the Region of Interest (ROI). A diagram of the parallelization model of the application is shown in Figure 12.

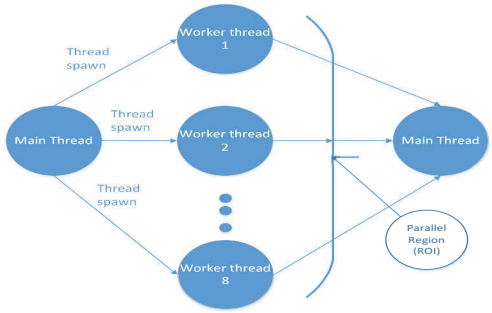

Fig. 12: Parallelization model of the benchmark

We execute the application with its native input set which has 10 million options and follow the subsequent approach for validation. From Figure 13 we choose performance groups and run Blackscholes in configurations defined in the groups. We measure the achieved performance of the parallel region (ROI) in number of options calculated per second.

The goal of the validation is to verify the relative energy efficiency among configurations inside one performance group is preserved for the application case study. For validation purpose, we select the last 3 groups of configurations from Figure 13, those having the highest performance level from our model. We then execute the Blackschole application and measure the corresponding power dissipation and performance for the six selected configurations. The measured performance levels achieved by different configurations within a group did not vary more then 10%. The resulting energy efficiency is

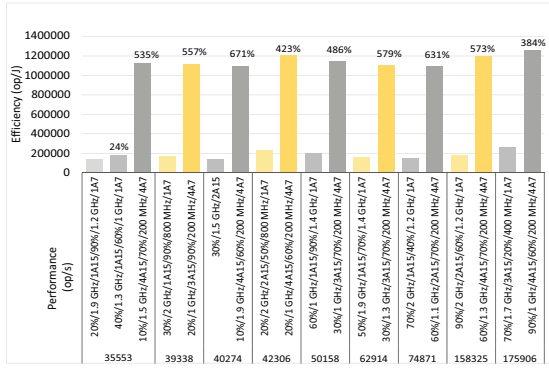

Fig. 13: Performance vs. efficiency levels

shown in Figure 14. In addition Figure 14 shows the achieved

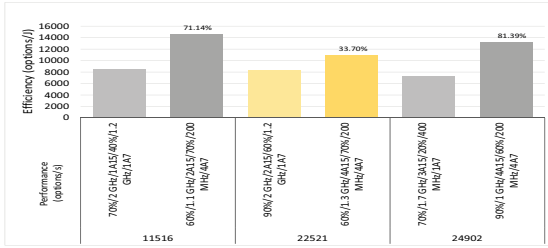

Fig. 14: Performance vs. efficiency for blackscholes

average performance levels for each configuration. The relative efficiency difference within a configuration group is shown. The differences in efficiency range from 33% to 81%. As we can see from the comparison of Figures 13 and 14 our model is able to indicate the most efficient configuration for each group. The differences in percentage between Figures 13 and 14 can be explained by the large number of memory instructions present in Blackscholes compared to the synthetic load generator Spurg-bench.

To further evaluate our approach, we compare the efficiency of few configurations with and without using Sched_dealine. Figure 15 shows the differences in efficiency levels when using the utilization rate as a configuration parameter. We can see that by taking into account the utilization rate we can achieve better efficiency.

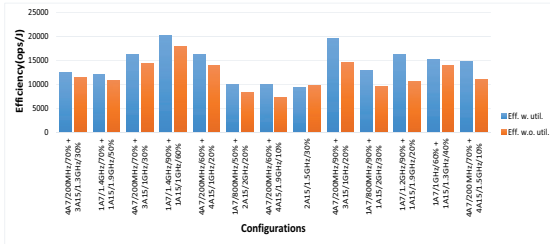

Fig. 15: Comparison of efficiency with and without utilization factor

VII. Conclusions and Future Work

In this paper we analyzed the achievable energy gains when exploiting at the same time heterogeneity, voltage and frequency scaling and utilization rate control techniques. We proposed a approach to build an energy efficiency model and

table based on platform configurations. We analyzed the energy efficiency variation for different platform configurations providing the same level of performance. We show that trading the number and type of core with frequency and voltage level and core utilization rate can lead to substantial energy gain.

As future work we see the possibility of building a more general model based on a better mix of instructions, giving the possibility to represent a larger variety of real world applications. We plan to consider the possibility of a runtime system, which will divide the execution of an application in phases. For each phase the appropriate architecture configuration can be established, and selected at runtime. This will provide an energy efficient runtime system at the granularity of application phases. In this work we considered the utilization at cluster level. It would be interesting to explore the impact of using different utilization rates inside a cluster, and study the trade-off between configuration space complexity and resulting energy gain.

ACKNOWLEDGMENT

This research was supported by a grant from the Erasmus Mundus EUROWEB+ Scholarship Programme.

REFERENCES

[1] Kenzo Van Craeynest, Aamer Jaleel, Lieven Eeckhout, Paolo Narvaez, and Joel Emer. Scheduling Heterogeneous Multi-cores Through Performance Impact Estimation (PIE). In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, pages 213–224, Washington, DC, USA, 2012. IEEE Computer Society.

[2] Yangchun Luo, Venkatesan Packirisamy, Wei-Chung Hsu, and Antonia Zhai. Energy Efficient Speculative Threads: Dynamic Thread Allocation in Same-ISA Heterogeneous Multicore Systems. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, PACT '10, pages 453–464, New York, NY, USA, 2010. ACM.

[3] Andrew Lukefahr, Shruti Padmanabha, Reetuparna Das, Ronald Dreslinski, Jr., Thomas F. Wenisch, and Scott Mahlke. Heterogeneous Microarchitectures Trump Voltage Scaling for Low-power Cores. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, PACT '14, pages 237–250, New York, NY, USA, 2014. ACM.

[4] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel. Mapping on multi/many-core systems: Survey of current and emerging trends. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–10, May 2013.

[5] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, October 1974.

[6] M. B. Taylor. A landscape of the new dark silicon design regime. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–1, March 2014.

[7] D. H. Woo, D. H. Woo, D. H. Woo, D. H. Woo, H. H. S. Lee, H. H. S. Lee, H. H. S. Lee, and H. H. S. Lee. Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era. *Computer*, 41(12):24–31, December 2008.

[8] E. S. Chung, P. A. Milder, J. C. Hoe, and K. Mai. Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs? In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 225–236, December 2010.

[9] Sparsh Mittal and Jeffrey S. Vetter. A Survey of CPU-GPU Heterogeneous Computing Techniques. *ACM Comput. Surv.*, 47(4):69:1–69:35, July 2015.

[10] ARM big.LITTLE white paper.

[11] Connor Imes and Henry Hoffmann. Minimizing Energy Under Performance Constraints on Embedded Platforms: Resource Allocation Heuristics for Homogeneous and single-ISA Heterogeneous Multi-cores. *SIGBED Rev.*, 11(4):49–54, January 2015.

[12] E. Nogues, M. Pelcat, D. Menard, and A. Mercat. Energy Efficient Scheduling of Real Time Signal Processing Applications through Combined DVFS and DPM. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 622–626, February 2016.

[13] Juri Lelli, Claudio Scordino, Luca Abeni, and Dario Faggioli. Deadline scheduling in the Linux kernel. *Software: Practice and Experience*, 46(6):821–839, June 2016.

[14] F. Hällis, S. Holmbacka, W. Lund, R. Slotte, S. Lafond, and J. Lilius. Thermal influence on the energy efficiency of workload consolidation in many-core architectures. In *2013 24th Tyrrhenian International Workshop on Digital Communications - Green ICT (TIWDC)*, pages 1–6, September 2013.

[15] E. Del Sozzo, G. C. Durelli, E. M. G. Trainiti, A. Miele, M. D. Santambrogio, and C. Bolchini. Workload-aware power optimization strategy for asymmetric multiprocessors. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 531–534, March 2016.

[16] S. Holmbacka, E. Nogues, M. Pelcat, S. Lafond, and J. Lilius. Energy efficiency and performance management of parallel dataflow applications. In *2014 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pages 1–8, October 2014.

[17] D. Liu, J. Spasic, G. Chen, and T. Stefanov. Energy-efficient mapping of real-time streaming applications on cluster heterogeneous MPSoCs. In *2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, pages 1–10, October 2015.

[18] Santanu Sarma, T. Muck, Luis A. D. Bathen, N. Dutt, and A. Nicolau. SmartBalance: A Sensing-driven Linux Load Balancer for Energy Efficiency of Heterogeneous MPSoCs. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 109:1–109:6, New York, NY, USA, 2015. ACM.

[19] Arunachalam Annamalai, Rance Rodrigues, Israel Koren, and Sandip Kundu. An opportunistic prediction-based thread scheduling to maximize throughput/watt in AMPs. In *Proceedings of the 22nd international conference on Parallel architectures and compilation techniques*, pages 63–72. IEEE Press, 2013.

[20] Thannirmalai Somu Muthukaruppan, Anuj Pathania, and Tulika Mitra. Price Theory Based Power Management for Heterogeneous Multi-cores. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 161–176, New York, NY, USA, 2014. ACM.

[21] Simon Holmbacka, Frederic Hallis, Wictor Lund, Sébastien Lafond, and Johan Lilius. TUCS Publication Database: Energy and Power Management, Measurement and Analysis for Multi-Core Processors. Technical report.

[22] Christian Bienia and Kai Li. Parsec 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, volume 2011, 2009.

Paper 2

Exploring Energy Efficiency Model Generalization on Multicore Embedded Platforms

2

Hergys Rexha, Sébastien Lafond

Originally published *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*

Exploring Energy Efficiency Model Generalization on Multicore Embedded Platforms

***Abstract*—In this paper we investigate the relation between energy efficiency model and workload type executed in modern embedded architectures. From the energy efficiency model obtained in our previous work we select a few configuration points to verify that the prediction in terms of relative energy efficiency is maintained through different workload scenarios. A configuration point is defined as a set of platform tunable metrics, such as DVFS point, DPM level and utilization rate. As workloads, we use a combination of synthetic generators and real world applications from the embedded domain. In our experiments we use two different architectures for testing the model generality, which provide examples of real systems. First we have a comparison of the efficiency obtained by the two architecturally different chips (ARM and INTEL) in different configuration points and different workload scenarios. Second we try to explain the different results through the thermal management done by the two different chips. At the end we show that only in the case of workloads highly composed by integer instructions the results from the two architectures converge and show the need for a specific model trained with integer operations.**

I. Introduction

Energy consumption is a key issue in today's electronic systems, ranging from IoT nodes to server farms. Information technology has acquired a big part of our everyday life, requiring a large amount of energy. The number of computing systems we use today has never been so high in the past, and is rapidly increasing [1]. The current energy production figures show that in Europe, only a small part of the energy required is produced from renewable sources, relaying mostly in fossil fuels, with a related strong impact in the environment in terms of GHG (green house gasses) emissions. Recently the world meteorological organization in its buletin said that the levels of CO_2 in 2016 had an unprecedented increment [2], giving an alarm about the levels of greenhouse gases and the resulting climate changes. As Europe has always been a leader in policies for environmental protection and carbon reduction initiatives, in the future guidelines, there is a high pressure to increase the energy efficiency of electronic devices [3]. No matter if we consider a data center or mobile devices, the imperative is still the same: a decrease in energy consumption is nedded. Depending from the computing domain, approaches have been long proposed for achieving reduction in the energy consumption. One of the largest markets in electronic devices which has had the fastest rise in the past years, is the mobile systems domain. Exposed to a huge number of use cases, mobile devices face different requirements which often trade-off with low energy consumption. The most obvious requirement is performance, which affects directly the power dissipation of mobile systems. Today's processor chips are reaching levels of performance which are able to cope with the most performance hungry applications. In the future, different applications like virtual reality, artificial inteligence and machine learning will increase the level of performance required from processor chips. The problem in this scenario is that we need also to be efficient. Recent approaches from the industry try to achieve better energy efficiency through the use of heterogeneous systems, which enclose different computing elements inside a single chip [4]. Recently, industry proposes an increased level of heterogeneity present on a multiprocessor system on chip (MPSoC) with approaches like [5] and [6] where we go beyond the idea of a two cluster heterogeneity, by adding another cluster of cores considered as middle level performance, obtaining a tri-cluster heterogeneity. In this way authors promise to cut power dissipation by 50%[1]. Also, from ARM we have the latest technology named DynamIQ [2] providing many options for organizing high performance and energy efficient cores inside a cluster. In this paper we follow the work done in [7], which builds an energy efficiency model based on platform configuration points. Platform configuration points are combinations of available actuators present in today's heterogeneous architectures. In the next section, we will present the ideas behind this work and the questions raised which we try to answer in this paper.

A. Why we did this work?

In our previous work we experimentally build an energy efficiency model for two widely used ARM core types which compose the ARM big.LITTLE architecture (Cortex A-15 and Cortex A-7). The energy efficiency model is based on synthetic workload composed mostly from floating point instructions. In the following steps we would like to validate the generality of the model with different instruction mixes and verify that the relative efficiencies of the model points are still valid. The main research questions that we try to address in this work are the followings:

1) Can we have a general energy efficiency model, without looking at the type of load/instructions?
2) Are the relative energy efficiency values for different configuration points kept for different instruction mixes?

In our definition of platform configuration point we use configurable actuators available in today's platforms. In heterogeneous platforms they are defined by:

- The number and type of cores to utilize for computations
- The frequency each type of core can have
- The utilization rate to be used by each type of core

[1]http://www.mediatek.com/products/smartphones/mediatek-helio-x30
[2]http://developer.arm.com/technologies/dynamiq

The combination of the previous parameters defines a platform configuration point. From the energy efficiency model in [7] we derive a lookup table composed of all the configuration points available on a given platform and the related performance and energy efficiency values they provide.

Furthermore, in our experiments we use another architecture to compare energy efficiency results achieved by platform configuration points in different workload types. We choose two sets of configuration points to be used during the experiments. In the ARM architecture, the selected platform configuration points are described in Table I. As a second architecture we choose Intel Atom, and define the configuration points presented in Table II. For both architectures, in the four configuration points chosen we use the maximum executing parallelism available and a mix of choices when applying or not, utilization control and DVFS. We enforce utilization control on a particular thread by means of a real time type scheduler, which is named *sched_deadline*. For more information on the methodology used and the details of the configurations, refer to [7]. With utilization control, which is expressed in percentage, we select the load level the core will reach while running the computations. Beside the description of each configuration point, the resulting performance and efficiency values are reported in the tables. These values originate from the energy efficiency model in [7]. Our investigations for the previous questions lead us to the forthcoming questioning:

3) Do ARM and Intel architectures provide the same energy efficiency?
4) Do the thermal characteristics play a role in the efficiency of the configurations, especially when controlling the load?

We will try to answer these questions through a wide set of experiments as they will be presented in section 3.

II. Related work

There are not many works who analyse the relation between the composition of the workload being executed and the right choice to execute it in a highly energy efficient way. The closest to this topic is the work in [8] where the authors show that by taking into account the mix of instructions from the workload, an energy efficient mapping could be done in a heterogeneous systems. By knowing which core type is best for a certain workload the scheduling decisions could be taken in such a way to achieve high levels of energy efficiency. The authors promise to save energy in the interval 7.1% to 31.3% if a workload-aware scheduler will be used. In contrast with their work we want to validate the results obtained before with our energy model, in the context of different workload type. In our second test case we use real world applications which represent embedded applications. In [9], authors try to answer the question whether ISA plays a significant role in the energy efficiency of different architectures. They analyze three architectures with different workload types with the conclusion that there is nothing more energy efficient in one ISA compared to another. In [10] authors investigate the effects of static power consumption on the energy efficiency of two popular embedded processors like ARM and Atom. For different benchmark types they show that if the right level of static power is present in the processor (from the design phase), than by choosing the right thread level parallelism, better energy efficiency could be achieved. In contrast with this work we measure from the experiments the static part of the dissipated power and discuss the effect of the static power on the energy efficiency differences between two popular embedded architectures such as Intel and ARM. We show that there is a significant difference in the static power between the two architectures when full parallelism is used. We believe that this difference has a strong impact on the results related to the energy efficiency. In [11], authors propose a runtime system which manages several workloads by taking into account performance requirements and application variability, with the goal of achieving better energy efficiency. The results claim to improve energy efficiency by 33% compared to existing approaches. They use the number of memory references per instruction as a key metric for classifying the workloads in three groups. Based on this behaviour they optimize the mapping process. This work considers only heterogeneity and DVFS as basic actuators for energy efficiency. In contrast in this work we consider the possibility to use core level utilization as an additional feature which can produce benefits in terms of energy efficiency. We use the relative presence of simple instruction types as a characterization of the workload.

III. Experimental settings

In this section we will present the hardware and software tools that we used during our experiments and also the used measurement framework.

A. Hardware platforms

We use two hardware platforms in these experiments. A summary of their characteristics can be found in Table III.

B. Measurement framework, interval definition with the oscilloscope

The ODROID board is installed with Linux Ubuntu 14.04, kernel 4.2.0, GCC 4.9 while the UP board is installed with Ubilinux 3.0, kernel 4.4.0 GCC 4.9. While performing the experiments on both boards the minimal services of Linux system are running. For power measurements we use an external power supply with a current/power IC monitor [12]. The supply voltage is maintained stable at 5V, and the power data are logged with an interval of 10 ms, in order to have a good trade-off between accuracy and resolution of the logged data. To investigate on the effects of chip temperature on the power dissipation we use a two channel PC oscilloscope, PicoScope 2205 [13], with one channel connected to the board power supply rail, measuring the current consumption, and the other channel used to measure the voltage level of a selected GPIO pin from the board. We use that pin as a synchronizing START/STOP signal for collecting power data. We run the workloads in a back to back fashion and during this time we collect the power data for each separate run. The performance

TABLE I
PLATFORM CONFIGURATION POINTS USED IN THE EXPERIMENTS WITH ARM ARCHITECTURE

Name	Configuration	Performance (op/s)	Energy Efficiency (op/J)
C1	4A15/1.1GHz/100% + 4A7/0.6GHz/100%	217433	900442
C2	4A15/2GHz/100% + 4A7/1.4GHz/100%	377885	572154
C3	4A15/1.1GHz/60% + 4A7/0.6GHz/60%	154912	1010780
C4	4A15/2GHz/60% + 4A7/1.4GHz/60%	284522	657773

TABLE II
PLATFORM CONFIGURATION POINTS USED IN THE EXPERIMENTS WITH INTEL ARCHITECTURE

Name	Configuration	Performance (op/s)	Energy Efficiency (op/J)
C1	4 Intel Atom/1.1GHz/100%	207641	105669
C2	4 Intel Atom/1.92GHz/100%	208550	106566
C3	4 Intel Atom/1.1GHz/60%	106326	164171
C4	4 Intel Atom/1.92GHz/60%	120481	119743

TABLE III
SUMMARY OF THE USED PLATFORMS

	Intel	ARM	
Architecture	x86	ARM v.7	ARM v.7
Processor	Atom z8350	Cortex-A7	Cortex-A15
Cores	4	4	4
Frequency	1.92 GHz	1.4GHz	2.0GHz
Width	2-way	2-way	3-way
Issue	OoO	In Order	OoO
L1 Data	32 KB	32KB	32KB
L1 Instr	24KiB	32KB	32KB
L2	2MB	512KB	2MB
Memory	4GB DDR3L	2GB LPDDR3	
SIMD	AVX	NEON	NEON
Tech Node	14nm	28nm	28nm
Platform	DevBoard	DevBoard	DevBoard
Products	UPBorad	ODROID XU4	ODROID XU4

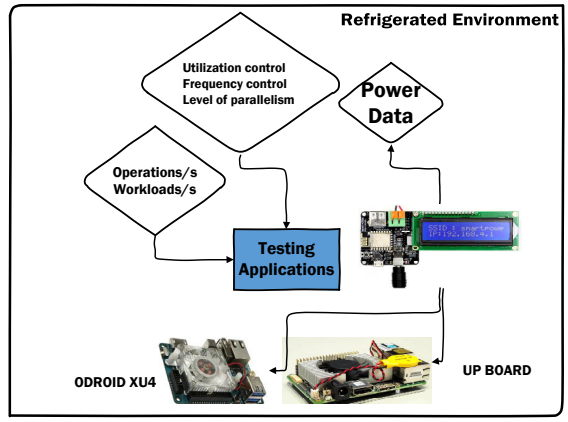

Fig. 1. Diagram of experiment infrastructure.

data are logged in a different file in terms of instructions per second or workloads per second. For more accurate results, each experiment is repeated 5 times and the average values of performance and power are reported. A diagram of the experimental framework is showed in Figure 1.

C. Presentation of the benchmarks: type of loads, synthetic vs. real

We used two categories of workloads for our experiments. The first category is composed of synthetic instruction mix, which are obtained from a synthetic workload generator called epEBench [14]. This benchmark uses several funcion models (Table IV) which use simple type of instructions. The benchmark is designed to integrate synthetic workload with analytical approach in analysing the energy efficiency of multi-core systems. In our experiments, we utilize an important parameter from the benchmark, which is the ability to control the workload-level executed by the processing core. This behavior is defined as the utilization rate in a real applications [14]. Furthermore with this benchmark we are able to simulate different type of instruction mixes through the use of function models and by means of the Linux pthread library we can select a specific level of parallelism.

The second workload category is composed of real world applications related to the embedded system domain. These applications are selected from CoreMarkPro[3] benchmark suite. It offers real-world examples of applications with different instruction mix. The workloads in CoreMarkPro are divided in two main categories: floating-point and integer. In our experiments we choose 4 workloads, 2 from each type. A description of their composition is made in Figure 2. The workloads used are:

- Linear Algebra workload which is a mathematical solver of equations through the Gaussian elimination method.
- FFT Radix 2 workload performs transformations with Radix2 on the input.
- XML parser workload parses an XML string and creates an ezxml structure with subsequent final validation of the

[3]http://www.eembc.org/coremark/index.php?b=pro.htm

TABLE IV
SET OF CURRENTLY USED LOAD FUNCTIONS FROM EPEBENCH.

Nr.	Load Function	Inst. Type	Description
1	run_dmul64_SIMD	muldiv (SIMD)	Double precision mult.
2	run_smul32_SIMD	muldiv (SIMD)	Single precision mult.
3	run_dsub64_SIMD	addsub (SIMD)	Double precision sub.
4	run_dmul	muldiv	FP multi.
5	run_dadd	addsub	FP addition
6	run_imul	muldiv	Fixed point mult.
7	run_iadd	addsub	Fixed point addition
8	run_branch	branch	Branch
9	run_imem	mem	Memory access (int)

results.

- Secure Hash Algorithm (SHA256) workload is composed from a subset of cryptographic hash functions with digests of 256 bits.

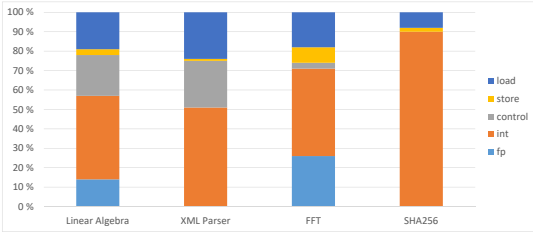

Fig. 2. Instruction mix of the chosen workloads from CoreMarkPro.

D. Temperature experiments

We have conducted experiments to measure the relation between power dissipation and chip temperature during workloads execution. The workloads from CoreMarkPro benchmark were run one after each other and the temperature of the chip was measured at 10 ms intervals. At the same time, the power data were logged with the external meter. With the collected data we can plot the relations between power and chip temperature during the execution time. Because of the large power dissipation during the execution of some of the workloads, especially in the Exynos 5422 chip, all the experiments concerning CoreMarkPro benchmark were run in a refrigerated environment with a controlled temperature of -18°C. Also the development boards were equipped with a fan powered with an external supply. With this infrastructure we were able to perform experiments without reaching the critical temperature of the core, where if so happens, the system will be turned off in order to prevent physical damages of the silicon.

IV. RESULTS

In this section we present some of the results which shed some light over the questions presented in the first sections. We first explore the generality of the energy efficiency model with regard to the executed workload. Then, through the synthetic benchmark, we value the impact of utilization control on the energy efficiency of different instruction types. At the end, a comparison of the two architectures regarding energy efficiency and a possible explanation for their difference will be discussed.

A. (Answer to question 1): The impact of workload type on the relative energy efficiency of the configuration points

We run the selected workloads from the CoreMarkPro benchmark at the highest degree of parallelism offered from the chosen platforms. The workloads are executed with 1000 iterations and the throughput is collected in terms of workloads per second as a performance metric. The average power dissipation is measured during execution time. In Figure 3 we show the energy efficiency results from the Exynos chip with the real world workloads for the selected platform configuration points. In general for the first three workloads there is not a significative difference in the efficiency provided by C1 and C3 configuration points. We remind that the only difference between these configuration points is the utilization control which is enforced in C3 at 60%, while in C1 the core reaches 100%. Both of these configurations set the cores at their middle level frequency (1GHz for A15 and 600MHz for A7) choosing a relaxed execution strategy or otherwise called the pace-to-idle approach where the execution is set at the lowest possible speed while still keeping the performance requirements. The difference although in energy efficiency remains high with C2 and C4 which enforce the race-to-idle strategy, or running at the fastest speed. Demonstrating that going at the fastest speed is not energy efficient. Remarkably during the cryptographic function execution we have different results of relative energy efficiency for the platform configuration points. Here, C1 and C2 show better energy efficiency than C3 and C4. This behaviour is against the model predictions.

The results for Intel are presented in Figure 4. Here, in the linear equation solver configuration C3 provides the best efficiency result, which is in accordance with the model (Table II), and the others three provide almost the same level of efficiency. We observe same behaviour with minor changes in the xml parsing workload and also in the FFT application. In the Secure Hash Algorithm still the best efficiency of the group is achieved by the C1 and C2 configurations, results that are the opposite compared to Table II. Apparently in the case of a strong presence of integer instructions in the executed workload, the actual model is not able to select the highest energy efficiency configuration point.

B. (Answer to question 2):Impact of load control on efficiency

We want to answer the question of whether the utilization control has an impact on efficiency, and evaluate if this impact is more important in some type of workloads rather than others. We use epEBench multi-core energy benchmark for generating workloads based on the function models presented in Table IV. The workloads are executed through the set of previously described configuration points (C1-C4). The performance data are collected in terms of instructions per second and the power data are logged through the external

meter. The results are presented in Figures 5, 6 and 7. According to the results for the ARM platform, C1 and C3 provide almost the same level of energy efficiency through all the model types used. The same happens between C2 and C4. In the case of workloads composed by integer additions and multiplications we cannot notice the behaviour observed from SHA256 workload in Figure 3.
In Intel (Figures 6 and 7), configuration C3 provides a slight better energy efficiency compared to C1, C2 and C4 which are together at the same level. While again as in ARM this observance is repeated through all the model used. Apparently for both architectures there is no significant difference in the energy efficiency levels achieved for different instruction types. Also, utilization control has the same effect independently of the instruction type executed.

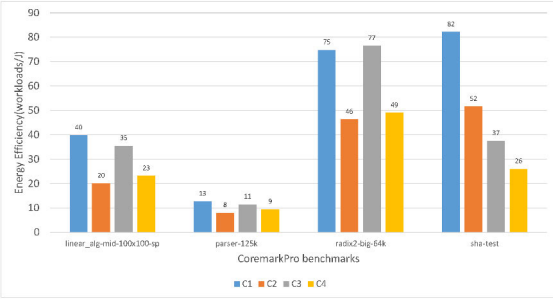

Fig. 3. Energy efficiency for the Exynos 5422 SoC workloads from CoreMarkPro

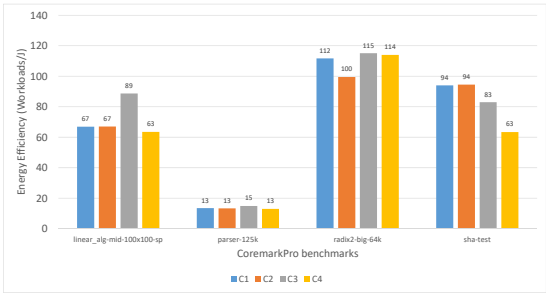

Fig. 4. Energy efficiency for the Atom x5 SoC with workloads from CoreMarkPro

C. (Answer to question 3):ARM vs. Intel, which architecture provides better energy efficiency?

This question is highly debated in the academic and industrial community circles. In our experiments we tackle both sides of the problem with a double scenario. On one side we use real applications from the embedded domain to measure the energy efficiency levels achieved in both architectures, on the other side we test again the architectures by using an energy benchmark which enables us to define the workload model to execute. In the experiments conducted with the real world applications, Intel succeeded to be more energy efficient than ARM in all the workloads. Even in the XML parser which shows the lowest scores in terms of energy efficiency. A possible explanation for low scores is the poor ILP present in the application, versus the highest ILP of the workload set found in the FFT workload which shows the highest scores in energy efficiency. A possible explanation for the difference between ARM and INTEL can be found by the presence of more static power dissipated in the Exynos chip. From the graphs in Figures 8 and 9 we observe the relation between power dissipated and chip temperature during the execution of Linear Algebra workload in configuration C1 and C3. Here, for ARM we can see a positive slope in the temperature curve for the two configuration points, the slope is more emphasized in configuration C1 with higher temperatures reached at the end of execution. This observation guides us to the conclusion that more static power is present in the power data, so even though ARM shows better performance (more cores present in chip), still energy efficiency scores are in favour of Intel Atom. As a comparison we have conducted the same experiments in the INTEL architecture and measured Atom z-8350 temperature and power with the same workload. The results are plotted in Figures 10 and 11. We can notice in the graphs that the temperature of the chip reaches a steady state from the beginning of the experiment and remains constant through the execution, leading to less static power dissipation.

A total different picture is obtained in the second case, with different function models used in the epEBench synthetic benchmark. ARM architecture provides better energy efficiency in almost all the models used, except for branches, which show better results in Intel. This could be related to the better branch predictor present in the Atom processor.
Furthermore in Intel, there is not much difference in efficiency between the single point and double point operations, while in ARM single precision operations are more energy efficient than double precision. We believe this is due to the presence of heterogeneity in ARM architecture, with the A7 cores providing increased levels of efficiency. Also, the energy efficiency achieved in the workloads composed by SIMD instruction is higher in ARM than Intel, understandably if we consider the presence of NEON execution engine in the ARM platform.

D. (Answer to question 4): Thermal characteristic influence on the energy efficiency results

From the temperature experiments conducted in the previous subsection we could notice the presence of an increased power dissipation in ARM compared to Intel, leading to a difference in energy efficiency scores. The previous experiments were conducted in a highly refrigerated environment, as a result a controlled rise in chip temperature. In a second experiment we remove the boards from the controlled environment and try to execute again the Linear workload with 1000 iterations and with only the active fan as a cooling system. We monitor the current consumed during the execution with an oscilloscope.

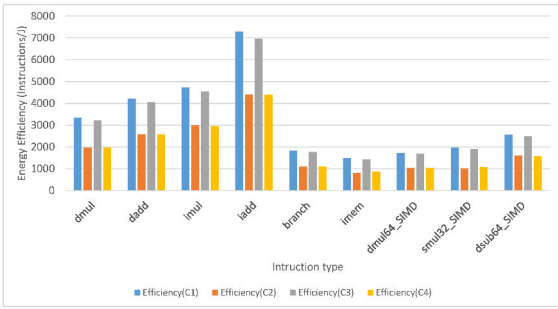

Fig. 5. Energy efficiency for the Exynos 5422 SoC with function models from epEBench

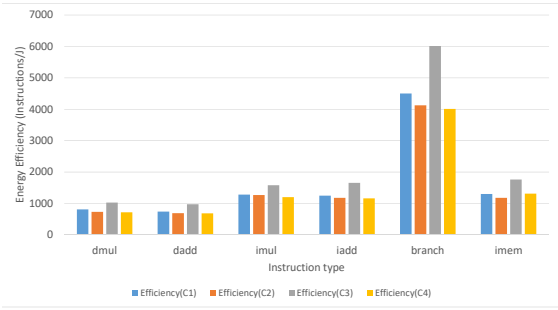

Fig. 6. Energy efficiency for the Atom x5 8350 SoC with function models from epEBench

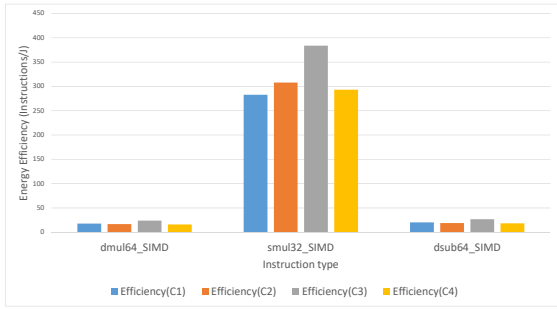

Fig. 7. Energy efficiency for the Atom x5 8350 SoC with function models from epEBench

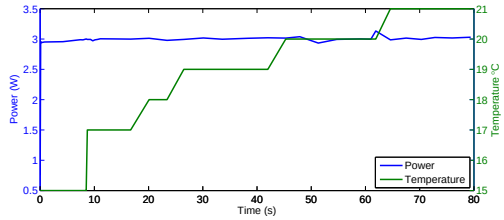

Fig. 8. Measurements on ARM with linear workload and C1 configuration point

In Figure 12 we can see the results obtained in ARM. The red line show the interval frame of the workload execution, before starting we can see an idle current consumption at an average level of 0.6A. After the start of the workload the current consumption reaches levels of 2.2A and while subsequent iterations of the workload are executed the current consumption grows until it reaches levels of 3.2A where the power dissipated reaches values that increase chip temperatures to critical values. As a safety measure the system is switched off, as we can notice that at the end of the execution frame the oscilloscope is recording 0A on the current consumption. Since the same workload is executed multiple times, the subsequent increase in power dissipation is attributed to the static power dissipated in the chip, with a value more than 5W associated with it.

The same experiment is conducted in Intel Atom with the result shown in Figure 13. The current consumption remains stable in the execution window which shows better management of the thermal effects with subsequent control of the static power dissipated.

On the other side, the execution of synthetic loads produces different levels of static power dissipated. We observe that only special type of instructions while executed inside a loop produce more static power than others. In Figure 14, we notice the gradual rise in current consumption while executing a loop of memory instructions. This increase is associated with the continuous rise in power dissipated, due to the increment of static power part. If we compare with the INTEL platform, in Figure 15, there is not such perception of linear increase in power dissipated. For this workload such behaviour explains the similar results in energy efficiency values for the two platforms even though in ARM we have double number of cores with consequent better performance. Such behaviour is not present for all instruction types, for example while executing double precision multiplications thermal effects result in a more stable power scenario as we can see in Figure 16.

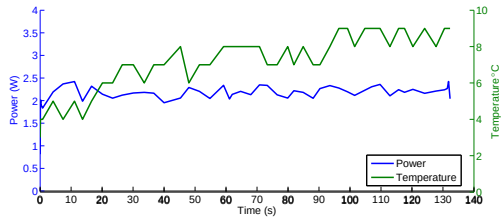

Fig. 9. Measurements on ARM with linear workload and C3 configuration point

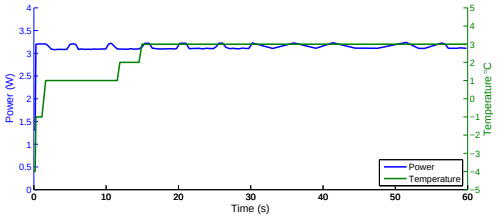

Fig. 10. Measurements on INTEL with linear workload and C1 configuration point

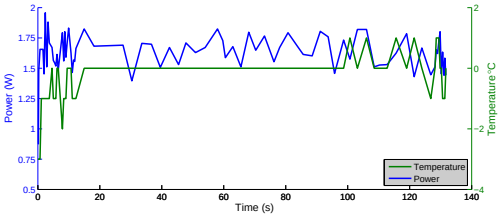

Fig. 11. Measurements on INTEL with linear workload and C3 configuration point

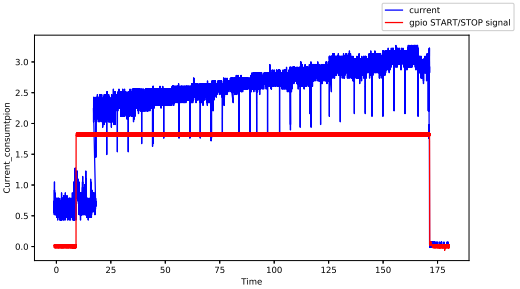

Fig. 12. Measurement of the current consumption during the execution of linear workload from the benchmark suite

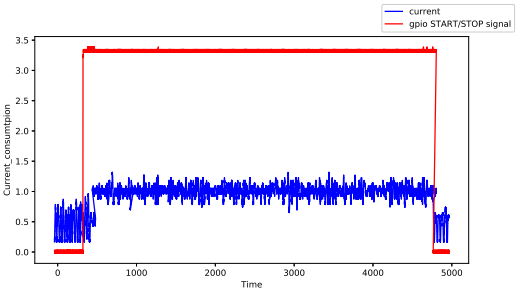

Fig. 13. Measurement of the current consumption during the execution of linear workload from the benchmark suite

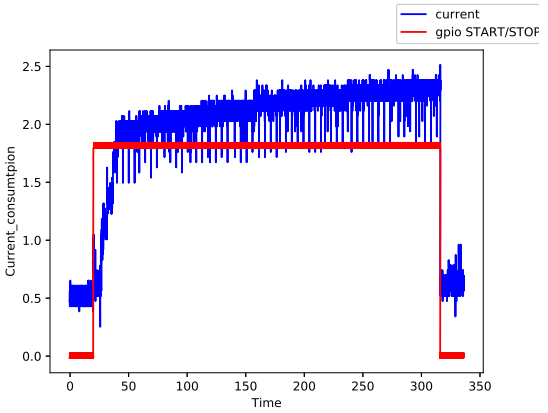

Fig. 14. Measurement of the current consumption during the execution of integer memory instructions in the ARM platform

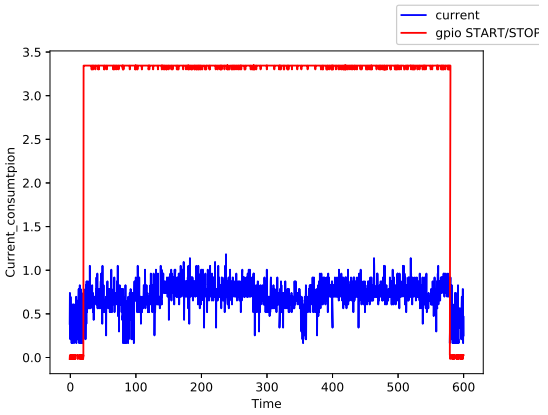

Fig. 15. Measurement of the current consumption during the execution of integer memory instructions in the INTEL platform

V. Conclusions

In this paper we answer questions regarding the generality of the previous obtained energy efficiency model [7], in the context of the embedded systems domain. We select two hardware platforms for running experiments, which are based on two popular embedded architectures such as ARM and INTEL. In the experiments with real applications from the CoreMarkPro suite, ARM and Intel show results approximately in accordance with the model values for workloads with evenly distributed instruction mixes. In case of workloads with high levels of integer instructions, we need another model to find high energy efficiency configuration points.

When we try to investigate the effect of utilization control on the energy efficiency with different instruction types, we use the multicore energy benchmark epEBench, which exe-

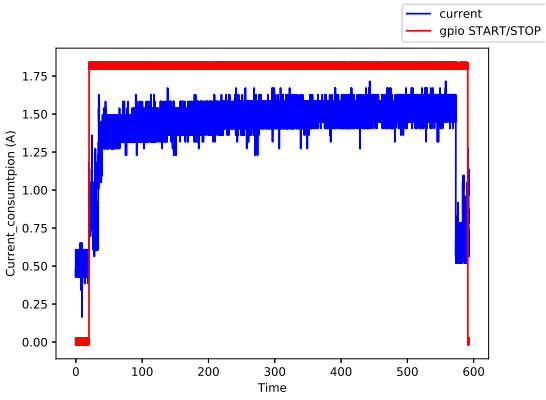

Fig. 16. Measurement of the current consumption during the execution of double precision instructions multiplication instructions in the ARM platform

cutes custom function models. We use 9 type of function models which test most common instruction types, with results in ARM that show no significant difference in the energy efficiency provided by utilization control on some special type of workload.

In Intel, utilization control seems to provide overall better energy efficiency results than in ARM, with workloads composed from branch instructions resulting more susceptible to a control load level in terms of energy efficiency.

Comparing the absolute results of energy efficiency between the two architectures produces a two faced picture. When using real applications as workloads Intel shows better efficiency levels than ARM in all the applications tested. We get a total different result when using synthetic workloads, where ARM shows better absolute values of energy efficiency in almost all the instruction types used. The performance provided by 8 cores in ARM is larger than the additional increase in power dissipated. Exception is made for branches and memory operations. With branches, Intel shows far more better efficiency than ARM. The performance with this type of load is almost the same in the two platforms, but less power is dissipated in Intel and we think this is due to a more efficient branch predictor in the Atom core. With purely memory load, we measure the same results in energy efficiency, even though in ARM the performance is much higher in terms of operations per second. The results are explained with a more stable level of power dissipated in Intel compared to ARM with this type of load. Another observation that can be made by results obtained from these experiments, is the limited efficacy of synthetic benchmarks when assessing properly energy efficiency comparison. This can be explained by the fact that static power effects are not strongly present while executing synthetic benchmark with only certain type of execution units stressed, as shown in section 4D, which explains the different energy efficiency results between synthetic and real workloads.

As a possible explanation of energy efficiency values is the difference of the two architectures in managing thermal effects. As we show by the experiments done with the oscilloscope Intel is better in managing thermal effects than ARM, which results in less static power present in Intel compared to ARM.

ACKNOWLEDGMENT

This research was supported by a grant from the Erasmus Mundus EUROWEB+ Scholarship Programme.

REFERENCES

[1] E. Gelenbe and Y. Caseau, "The Impact of Information Technology on Energy Consumption and Carbon Emissions," *Ubiquity*, vol. 2015, no. June, pp. 1:1–1:15, Jun. 2015. [Online]. Available: http://doi.acm.org/10.1145/2755977

[2] "World meteorological organization: Co2 levels surged to record high in 2016," https://www.cbsnews.com/news.

[3] "Europe's Energy Transition - 1st Edition." [Online]. Available: https://www.elsevier.com/books/europe-s-energy-transition/manuel-welsch/978-0-12-809806-6#utm_source=SciTech%20Connect&utm_medium=Energy%20New%20Releases&utm_campaign=STC317

[4] A. Holdings. (2013) big.little technology: The future of mobile making very high performance available in a mobile envelope without sacrificing energy efficiency. [Online]. Available: https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf

[5] H. T. Mair, G. Gammie, A. Wang, R. Lagerquist, C. J. Chung, S. Gururajarao, P. Kao, A. Rajagopalan, A. Saha, A. Jain, E. Wang, S. Ouyang, H. Wen, A. Thippana, H. Chen, S. Rahman, M. Chau, A. Varma, B. Flachs, M. Peng, A. Tsai, V. Lin, U. Fu, W. Kuo, L. K. Yong, C. Peng, L. Shieh, J. Wu, and U. Ko, "4.3 A 20nm 2.5ghz ultra-low-power tri-cluster CPU subsystem with adaptive power allocation for optimal mobile SoC performance," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, Jan. 2016, pp. 76–77.

[6] H. Mair, E. Wang, A. Wang, P. Kao, Y. Tsai, S. Gururajarao, R. Lagerquist, J. Son, G. Gammie, G. Lin, A. Thippana, K. Li, M. Rahman, W. Kuo, D. Yen, Y. C. Zhuang, U. Fu, H. W. Wang, M. Peng, C. Y. Wu, T. Dosluoglu, A. Gelman, D. Dia, G. Gurumurthy, T. Hsieh, W. Lin, R. Tzeng, J. Wu, C. Wang, and U. Ko, "3.4 A 10nm FinFET 2.8ghz tri-gear deca-core CPU complex with optimized power-delivery network for mobile SoC performance," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2017, pp. 56–57.

[7] H. Rexha, S. Holmbacka, and S. Lafond, "Core level utilization for achieving energy efficiency in heterogeneous systems," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, March 2017, pp. 401–407.

[8] S. Holmbacka and J. Keller, "Workload Type-Aware Scheduling on big.LITTLE Platforms," in *Algorithms and Architectures for Parallel Processing*, ser. Lecture Notes in Computer Science. Springer, Cham, Aug. 2017, pp. 3–17. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-65482-9_1

[9] E. Blem, J. Menon, T. Vijayaraghavan, and K. Sankaralingam, "ISA Wars: Understanding the Relevance of ISA Being RISC or CISC to Performance, Power, and Energy on Modern Architectures," *ACM Trans. Comput. Syst.*, vol. 33, no. 1, pp. 3:1–3:34, Mar. 2015. [Online]. Available: http://doi.acm.org/10.1145/2699682

[10] A. F. Lorenzon, M. C. Cera, and A. C. S. Beck, "On the influence of static power consumption in multicore embedded systems," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 1374–1377.

[11] B. K. Reddy, A. K. Singh, D. Biswas, G. V. Merrett, and B. M. Al-Hashimi, "Inter-cluster Thread-to-core Mapping and DVFS on Heterogeneous Multi-cores," *IEEE Transactions on Multi-Scale Computing Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[12] HARDKERNEL. (2016) smartpower2 power supply. [Online]. Available: http://wiki.odroid.com/accessory/power_supply_battery/smartpower2

[13] PicoTech. Picoscope 2205. [Online]. Available: http://logicanalysers.com/testequipment/picotech/pc_oscilloscope/picoscope_2203_2204_2205_user_guide.pdf

[14] S. Holmbacka and R. Mller, "epebench: True energy benchmark," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, March 2017, pp. 426–429.

Paper 3

Energy-Efficient Actor Execution for SDF Application on Heterogeneous Architectures

Hergys Rexha, Sébastien Lafond, Karol Desnos

3

Originally published *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*

Energy-Efficient Actor Execution for SDF Application on Heterogeneous Architectures

Hergys Rexha, Sébastien Lafond
Faculty of Science and Engineering
Åbo Akademi University, Turku Centre for Computer Science
Email:[hrexha],[slafond]@abo.fi

Karol Desnos
IETR, INSA Rennes, UBL, UMR CNRS 6164
Rennes, France
Email: kdesnos@insa-rennes.fr

***Abstract*—Heterogeneous systems promise to improve performance and endurance of power constrained systems, by utilizing computing elements of different power and performance characteristics. Such systems provide the possibility to trade number and types of core with Dynamic Voltage and Frequency Scaling (DVFS) levels and core utilization rate to achieve optimal energy efficiency. Therefore by making smart decisions on application scheduling and mapping we can exploit and maximize the benefits of using heterogeneous processors. At the same time, the application level of parallelism can conveniently be exposed by dataflow Models of Computation (MoCs). In this paper we show an energy efficient execution approach for heterogeneous architecture. We demonstrate the approach on a real-life streaming application modelled with Parameterized and Interfaced Synchronous Dataflow (PiSDF). The presented solution show how to integrate our approach in the workflow of a dataflow application prototyping tool. The results that we obtain demonstrate that, by using an optimal scheduling and mapping, more than 30% of energy reduction can be achieved on a single actor level.**

I. Introduction

The past years had seen a rapid development in the amount of data produced, processed and exchanged through computing systems, ranging from high end server farms to simple household devices. And the trend of technology seems to fuel even more this direction. Based on the data of electricity usage ascribed to Information and Communication Technologies (ICT) it is predicted that by the end of 2030 this sector will use as much as 51% of global electricity production. The ICT sector could be responsible for up to 23% of the globally released greenhouse gas emissions in 2030 [1]. These statistics confirm that we are living in an energy constrained world and in the future, if we dont increase the amount of energy generated from renewable sources, the impact on the environment will be tremendous.

It is therefore imperative to increase the energy-efficiency of computing devices. Energy efficiency in our days is a top level concern not only in data centers and supercomputing environments but also in embedded devices (handheld, IoT, wireless devices) which are steadily increasing in number [2]. In the past years industry witnessed a race towards producing computer chips with increased performance capabilities. Multi and many-core systems emphasized this trend and were adopted to satisfy the performance requirement from the application side. This led to an issue called dark silicon, where not all part of a chip can be powered at the same time [3]. Heterogeneous Multiprocessor System-on-Chip (MPSoC) appear as a probable solution for energy efficiency compared to homogeneous systems [4]. Especially single ISA heterogeneous architectures offer a good trade-off between energy-efficiency and programming efforts [5]. They consist of multiple computing elements with different power and performance characteristics, that share a single instruction set.

Finding the proper core diversity inside a MPSoC is a difficult task as suggested by E. Tomusk and co. in [6][7]. This is especially true in mobile applications where there are large workload variaties. Nowadays there are commercial versions available from different vendors like Samsung [8] and NVidia [9] which offer octa-core chips based on the ARM big.LITTLE architecture. The current market trend produces solutions with an increased level of heterogeneity. A recent example is the tri-cluster SoC provided by MediaTek Helio X30 [10], offering a deca-core over three clusters: one dual-core ARM Cortex-A73 up to 2.5GHz, one quad-core ARM Cortex-A53 up to 2.2GHz and one quad-core ARM Cortex-A35 up to 1.9GHz.

Achieving an energy efficient application execution on multicore heterogeneous architecture requires an overall view and deep understanding of the underlying software and hardware. First the software needs to expose the right amount of parallelism in order to utilize the capabilities of hardware, and then an efficient mapping of the software to hardware needs to be found. All of these objectives should be accomplished with respect of the application performance constraints.

Different Models of Computation (MoCs) have been used for modelling applications; Synchronous Dataflow (SDF) [11], Cyclo-Static Dataflow (CSDF) [12], Kahn Process Network (KPN) [13]. One of the main purpose of these MoCs is to expose parallelism inside the application. In general applications described by Dataflow Process Networks (DPNs) consist in a set of entities of which the application is composed, that are called actors. Actors communicate through FIFO buffers, an each actor can execute (fire) as soon as its input buffer contains enough data tokens (consumption rate). After execution the actor produces a certain number (production rate) of data tokens in its output buffer(s). The popularity of these models come from the simplicity in analysing the application structure.

In dataflow MoCs an application is defined as a graph of

concurrent tasks which communicate with each other through FIFO buffers. Synchronous Dataflow (SDF) [11] is probably the most used DPN, especially in signal processing applications. Inside a graph of actors the production and consumption rates of the actors are fixed numbers, which enable static analysis of the graph. Another valued property of SDF, not present in KPN, is the possibility to schedule the graph at compile time, as long as the graph is schedulable, deadlock-free and consistent. In SDF actors are stateless, which practically means that an actor can start several replicas of itself in parallel, if enough data tokens are available. With the exposed available parallelism, we can develop mapping techniques exploiting the benefits of using heterogeneous processors to achieve energy efficient execution of applications.

In this paper we demonstrate a mapping technique to achieve energy efficient application execution, integrated into the workflow of the Parallel and Real-time Embedded Executives Scheduling Method (PREESM) tool. PREESM is a development framework for rapid prototyping of dataflow applications. Applications developed with PREESM target execution in heterogeneous MPSoC, where the proposed approach defines the optimal platform configuration to run an actor under defined performance constraints. We define platform configurations by the following set of parameters:

- Level of parallelism achievable by an actor
- Number of cores to utilize
- Type of cores to utilize
- Dynamic Voltage and Frequency Scaling (DVFS) level of a cluster
- Cluster utilization rate

II. Related Work

Energy management in MPSoC has been studied with different perspectives. The largest part of studies relates to homogeneous multiprocessor system with voltage scaling at core level or at the system level [11], [14], [15]. In these works the general approach is to convert an original SDF graph to Homogeneous SDF (HSDF) in order to expose parallelism but with the shortcoming of increased memory usage and complexity. On the other hand approaches proposed in [16], [17] work directly on the level of SDF graph with the goal of energy minimization. They perform a design space exploration to find an energy efficient mapping for SDF applications on a homogeneous MPSoC with voltage scaling capability at the core level. In [15], authors propose an algorithm to find the core frequency level for a specific schedule and mapping of the SDF graph, for which the throughput constraints are met. In contrast to the aforementioned works, our focus is put on heterogeneous MPSoC, and consider ability to define voltage and frequency at the cluster level which has been shown to be a good trade-off [18], [19]. Moreover we use utilization level in addition to the number and type of core to define the platform configuration. We then use static schedules of SDF applications to demonstrate the energy efficiency of some platform configurations to run compute intensive actors. There are a number of works which also consider heterogeneous MPSoC as [20], [21]. In the first work authors consider a bin-packing algorithm for achieving particular load distribution which analytically is shown to minimize power dissipation. They do not consider using all cores at the same time losing the benefits provided by the energy efficient cores. In contrast our strategy takes into account the full computing potential of the platform with energy efficient and performance efficient cores altogether. In [21] authors consider clustered MPSoC, with the possibility of changing task allocation during runtime. By doing so, there is a high potential of large overheads spent in context switching, that can result in performance degradation. In contrast we define a static schedule based on configuration points that can be twice more energy efficient. The work which is more near to ours comes from Jelena Spasic and co. [22]. The authors propose and approach that determines the processor type and replication factor for each task of an SDF graph under throughput constraints. They try to balance the load across all computing cores in order to use frequency scaling as a main technique for achieving energy reductions. In contrast our approach uses DPM and utilization rate as additional parameters to gain even more energy reduction.

III. Short Presentation of Preesm

PREESM is an open-source rapid prototyping framework developed for research and educational purposes [23]. Rapid prototyping consists of extracting information from a system in the early stages of its development. It enables hardware/software co-design and favors early decisions that improve system architecture efficiency.

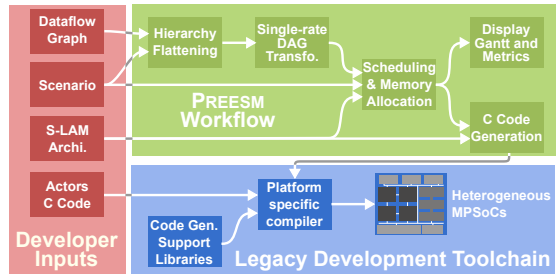

Fig. 1. Overview of PREESM workflow.

Figure 1 illustrates the different steps of the rapid prototyping workflow of PREESM. Inputs of the rapid prototyping workflow consist of: an algorithm model specified with a dataflow MoC, an architecture model respecting the System-Level Architecture Model (S-LAM) semantics [24], and a scenario providing mapping constraints and prototyping parameters. The scenario ensures the complete separation of algorithm and architecture models. The complete independence between the architecture and algorithm models simplifies the porting of an application on different targets.

In PREESM, the dataflow graph modelling the application first undergoes transformations in preparation for the rapid

prototyping steps. The main purpose of these graph transformation is to reveal all data, task, and pipeline parallelism of the application. Then, static multicore scheduling and memory allocation are executed to dispatch and order computations, communications and memory accesses to the different architecture elements. Finally, the multicore scheduling information is used to simulate the system behavior and to generate compilable code for the targeted architecture. Our approach is placed after the code generation step in PREESM workflow from Figure 1.

IV. Proposed approach

The proposed approach enables an energy efficient execution technique based on platform configuration points which offer high levels of efficiency. In order to define those platform configurations, we base our approach on power models built from experimental data as in [25]. With the related performance models we are able to construct energy efficiency models, which are the starting point of our energy minimization approach. We define optimal platform configuration points for running a particular actor with regard to it's performance level. The intention of this work is to show that our model can be efficiently used for deciding the schedule and mapping of dataflow applications. We demonstrate the usage of our strategy integrated in the workflow of a signal processing application development tool (PREESM). With today's architectures, which manage the execution of many short tasks regularly, the impact of this technique could result in large energy savings.

V. Experiment Setups

Our experiments intend to show that by setting the appropriate configuration for running a single actor from a dataflow application we will benefit in terms of energy compared to the schedule proposed initially by the tool.

A. Platform

We run our application on an ODORID XU4 development platform provided by HARDKERNEL company. The board is powered by an Exynos 5 MPSoC, which is an octa-core composed of 4 ARM Cortex A7 and 4 ARM Cortex A15 organized in a big.LITTLE architecture with Global Task Scheduling (GTS). The A7 cores can scale up to a frequency of 1.4GHz while A15 cores can reach a frequency of 2GHz. The development board is installed with Linux kernel 4.2. The application is compiled with gcc version 5.4 with optimization flag -02. For this platform, considering 9 utilization rates (from 10% to 90%) and 7 DVFS points, we reach total number of possible platform configurations equal to **64 008**.

B. Application overview

As an application case study we choose to use an SDF implementation of a stereo matching application from the computer vision application class [26]. The idea behind stereo matching is to compare input images taken by near cameras in order to produce as a result the depth of a scene. The algorithm used in this application offers a great opportunity for expressing parallelism which serves our approach based on parallel instances of an actor. The stereo matching application is composed of 12 actors and two tunable parameters which can be used for having a high degree of parallelism of the most compute intensive actors which are: *CostConstruction*, *AggregateCosts* and *ComputeWeights*. The parameters are *nbDisparity* and *nbOffsets*. The first represents the number of distinct values that can be found in the disparity map, while the second influences the size of the pixel area to be considered for the correlation calculus. For more information about the application refer to [27]. We choose to demonstrate our proposed approach on the *ComputeWeights* actor, which for each input pixel computes 3 weights based on the characteristics of neighbouring pixels.

C. Scheduling the application

As a proof of concept for our approach we want to use platform configuration points that provide high level of efficiency to execute a single actor of the *Stereo matching application*. In order to find high efficiency platform configuration points we build an energy efficiency table based on the power and performance models obtained in previous work [25]. The energy efficiency table lists platform configuration points ordered by their level of efficiency in a descending order. So at the top of the table we have high energy efficiency platform configuration points, which offer different types of scheduling and mapping options. In addition to the platform configuration name, in the energy efficiency table is shown the related performance and energy efficiency. For more detailed information on how the power and performance models see Table I [25]. From

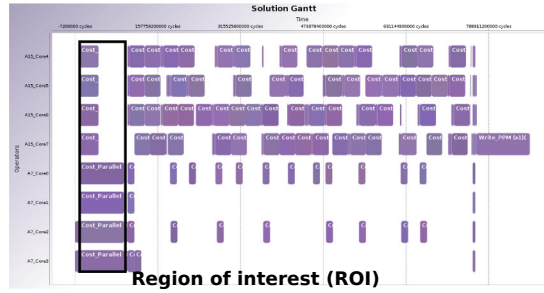

Fig. 2. Gantt chart of application schedule

the obtained energy efficiency Table I, we choose the highest energy efficiency platform configuration point, which in this case exposes the maximum level of parallelism (which for this architecture is 8). This most energy efficient platform configuration defines 4A15 cores clocked at 1.1GHz with a utilization rate at 60% and 4A7 cores clocked at 200MHz with a utilization rate at 60%. This platform configuration point is defined as the Configuration of Interest (COI). Therefore in the application schedule we will use 8 parallel instances of the *computeWeights* actor.

We focus on a certain region of the execution where *compute Weights* is run, as it is highligted on Figure 2. We

call this window of execution as our Region Of Interest (ROI). The platform COI is enforced using Linux *cpufreq* and *cpuidle* drivers for setting the frequency and the number of cores used. For controlling the utilization rate of the clusters we use sched_deadline [28] class of scheduler. We use a wrapper for executing the modified version of the actor in order to be able to enforce our COI for that particular region. A pseudo code of the wrapper is shown below.

```
1 #include<cpufreq.h>
2 #include<pthread.h>
3 computeWeightsWrapper(){
4 cpufreq_modify_policy_governor(core,"userspace");
5 cpufreq_set_frequency(core, FREQ);
6 pthread_create();
7 pthread_join();
8 }
```

With the wrapper code, in line 4 we set the frequency governor in *userspace* mode and in line 5 we set the frequency according to the COI. Lines 6-7 support the model of execution that we define for the modified actor code. The tool generates a C file for each processing element in the architecture. So for the architecture model used in PREESM, to represent octa-core CPU, the tool generates 8 files named according to the core they will execute. Each file contains calls to the actors executed on that core, inter core communications and synchronizations. Inside these files we insert our wrapper for setting the platform configuration for the defined ROI. The wrapper code forces that for each main thread executed on a core there is one additional thread that contains the code for the *computeWeights* actor as it is shown in Figure 3.

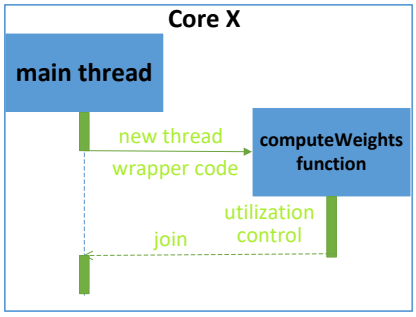

Fig. 3. Core computational model

TABLE I
ENERGY EFFICIENCY TABLE .

C($L_b/F_b/N_b/L_l/F_l/N_l$)	Perf.(op/s)	P(W)	Efficiency(op/J)
60%/1.1 GHz/4A15/60%/200 MHz/4A7	136 790	1.39	1 287 932
60%/1.1 GHz/4A15/80%/200 MHz/4A7	140 511	1.41	1 285 211
60%/1.1 GHz/4A15/70%/200 MHz/4A7	138 377	1.40	1 283 870
70%/1.1 GHz/4A15/60%/200 MHz/4A7	154 722	1.62	1 279 862
70%/1.1 GHz/4A15/80%/200 MHz/4A7	158 443	1.64	1 277 141
.	.	.	.
10%/2 GHz/1A15	8 000	0.18	42 962
10%/1.7 GHz/1A15	6 922	0.16	42 890
90%/2 GHz/1A15	72 028	1.99	36 165

D. Utilization control

To control the utilization rate in which to run a cluster of cores we use a new scheduling class, which is a Constant Bandwidth Scheduler, based on the Earliest Deadline First Algorithm. For implementing the control on the utilization rate, at a certain bandwidth, we use the following control code inside our actor.

```
1  struct sched_attr {
2   /* Size of this structure */
3   u32 size;
4   /* Policy (SCHED_*) */
5   u32 sched_policy;
6   /* Flags */
7   u64 sched_flags;
8   /*Nice value (SCHED_OTHER, SCHED_BATCH)*/
9   s32 sched_nice;
10  /*Static priority(SCHED_FIFO, SCHED_RR)*/
11  u32 sched_priority;
12  /* Remaining fields are for SCHED_DEADLINE */
13  u64 sched_runtime;
14  u64 sched_deadline;
15  u64 sched_period;
16 };
17 struct sched_attr attr;
18 ret = sched_getattr(0, &attr, sizeof(attr), 0);
19 if (ret < 0)
20 error();
21 attr.sched_policy = SCHED_DEADLINE;
22 attr.sched_runtime = runtime_ns;
23 attr.sched_deadline = deadline_ns;
24 ret = sched_setattr(0, &attr, 0);
25 if (ret < 0)
26 error();
```

By setting a certain runtime inside each period of actor execution we can keep a defined rate for running computations.

E. Measurements

We measure the power dissipation within the defined ROI, where *computeWeights* is run. In order to demonstrate our approach we need to verify the energy efficiency of our platform configuration point compared to running that specific actor at the highest performance platform configuration point. All the measurements will be done inside the interval of time in which *computeWeights* is executed. We collect 16k of samples for a window of 5s which means we collect power data every 0.3 ms.

For measuring power inside the ROI, we need a fast approach which records current consumed inside an interval that varies in execution time. For our input pictures, the smallest execution time is around 125ms. For setting the boundaries of our ROI we use board GPIO pins as a flag to measure the start and end time of the ROI. In this way we can have a precise measure of the time it takes for executing the section of the code in which we intend to apply the "Configuration of Interest (COI)" from our energy efficiency table. For measuring the power dissipation we use an oscilloscope which on one probe records the current drawn from the board and on the other senses the voltage on the GPIO pin which is used as flag for defining our ROI. The infrastructure is composed by several components as it is shown in the Figure 4. On the application side we set the load level, frequency point and amount of parallelism we want to achieve. We then set the mapping in

order that each core has to execute one actor replica. As a performance result we measure the execution time and number of frames per second obtained by the application. The power data are collected from a PicoScope 2205 oscilloscope.

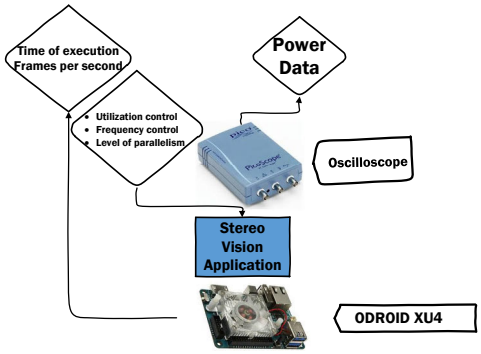

Fig. 4. Measuring infrastructure

VI. Results

We first measure the base board power dissipation. We remove from the board any external peripheral, including the fan, which is powered by an external supply. We connect to the board through the serial console in order to exclude ethernet port power. The results shown below represent the power at the CPU-Memory subsystem level. We measure power dissipation inside our ROI in three scenarios. In the first case we set our platform at the interested configuration point, in the second scenario we run at the highest performance platform configuration point. In the third case, as a middle point, we choose another platform configuration which provides a level of parallelism of 8 and according to the energy efficiency table has an efficiency level in between the previous points. In Figure 5 we show the power dissipation if we choose to use our COI for running the *computeWeights* actor. In Figure 6 we use the strategy to run everything at the fastest speed possible for both clusters. In order to have a further validation for the schedule we choose to run the application by setting *computeWeights* in the third platform configuration point. The power dissipation inside the ROI for the third platform configuration point is shown in Figure 7. The power graphs are composed by two running phases of the ROI. In phase A we have 8 running actor replicas of *computeWeights*, in phase B we are running only in the LITTLE cluster (the replicas on the big cluster have finished working). The fluctuations in power dissipation noticed mainly in Figures 5 and 7 are due to the utilization control inside the actor code. The constant bandwidth scheduler used in this case, alternates periods of computations to periods of sleeping.

The energy consumed by the *computeWeights* actor in the three scenarios is shown in Figure 8. As we can notice from the results, we consume less energy if the highest efficiency platform configuration point is used for our ROI. The second

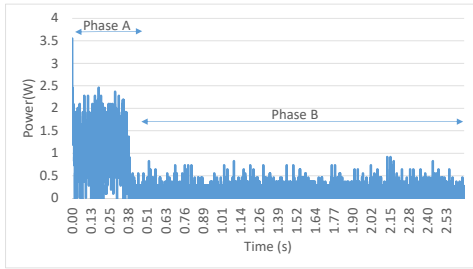

Fig. 5. ROI power for the energy efficient configuration

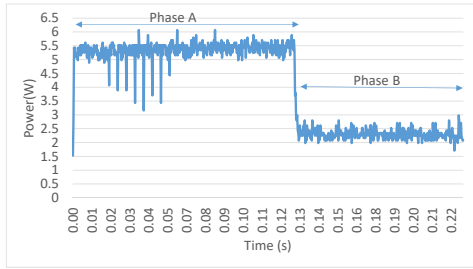

Fig. 6. ROI power for racing to idle strategy

best result for less energy consumed comes from the "middle" energy efficiency level platform configuration. The worst energy results are measured by the race-to-idle strategy which is the default schedule given by the tool. The difference between the corner cases of energy consumption differ by more than 30%.

VII. Conclusion and Future Work

In this paper we demonstrate the usage of an energy efficient execution approach on a signal processing application modelled with SDF. Our strategy focuses on a single compute intensive actor of the stereo matching application. The application is developed using PREESM tool. We propose a technique for integrating our energy efficiency model inside the workflow of PREESM. Through experimental results we show that the highest platform configuration point in the energy efficiency table provides an energy reduction of more than 30%, compared to the standard schedule proposed by the tool, within a single actor execution. By using a second platform configuration point from the energy efficiency table we show that the relative efficiency is preserved for the actor.

As a future work, we plan to extend our approach for the complete set of actors in the application taskgraph, with care of the performance aspect. We intend to validate the assumption that the relative difference in efficiency of platform configuration points is preserved for the application level.

Acknowledgment

This material is based upon work supported by an Erasmus Mundus scholar grant provided by the Euroweb+ project.

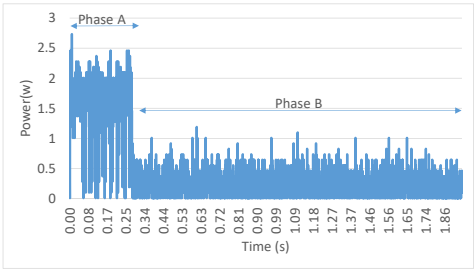

Fig. 7. ROI power for middle performance configuration

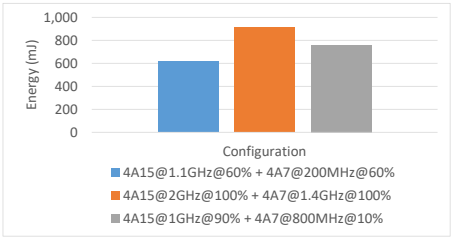

Fig. 8. energy consumed inside ROI

REFERENCES

[1] A. S. G. Andrae and T. Edler, "On Global Electricity Usage of Communication Technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, Apr. 2015. [Online]. Available: http://www.mdpi.com/2078-1547/6/1/117

[2] I. Insights, "Mobile handset IC market," http://www.electronicspecifier.com/around-the-industry/mobile-handset-company-forecast-increase-through-next-three-years, 2016.

[3] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 365–376, Jun. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024723.2000108

[4] S. Srinivasan, N. Kurella, I. Koren, and S. Kundu, "Exploring Heterogeneity within a Core for Improved Power Efficiency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1057–1069, Apr. 2016.

[5] T. Mitra, "Heterogeneous multi-core architectures," *IPSJ Trans. System LSI Design Methodology*, vol. 8, pp. 51–62, 2015. [Online]. Available: http://dx.doi.org/10.2197/ipsjtsldm.8.51

[6] E. Tomusk and C. Dubach, "Diversity: A Design Goal for Heterogeneous Processors," *IEEE Computer Architecture Letters*, vol. 15, no. 2, pp. 81–84, Jul. 2016.

[7] E. Tomusk, C. Dubach, and M. O'boyle, "Selecting Heterogeneous Cores for Diversity," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 4, pp. 49:1–49:25, Dec. 2016. [Online]. Available: http://doi.acm.org/10.1145/3014165

[8] Samsung, "Samsung Exynos 7," http://www.samsung.com/semiconductor/minisite/Exynos/w/solution/mobile_ap/7420/, 2016.

[9] "NVidia. Variable SMP { A Multi-Core CPU Architecture for Low Power and High Performance. Technical report, NVidia, 2011." [Online]. Available: http://www.nvidia.com/content/pdf/tegra_white_papers/tegra-whitepaper-0911b.pdf

[10] M. Inc, "MediaTek Helio x30," https://d86o2zu8ugzlg.cloudfront.net/mediatek-craft/documents/mediatek-helio-x30/Helio-X30-Product-Brief-PDFHX35PBA4-0224.pdf, 2017.

[11] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, Sept 1987.

[12] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cycle-static dataflow," *Trans. Sig. Proc.*, vol. 44, no. 2, pp. 397–408, Feb. 1996. [Online]. Available: http://dx.doi.org/10.1109/78.485935

[13] M. Broy, "A theory for nondeterminism, parallelism, communication, and concurrency," *Theoretical Computer Science*, vol. 45, pp. 1 – 61, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/030439758690040X

[14] A. Nelson, O. Moreira, A. Molnos, S. Stuijk, B. T. Nguyen, and K. Goossens, "Power minimisation for real-time dataflow applications," in *2011 14th Euromicro Conference on Digital System Design*, Aug 2011, pp. 117–124.

[15] M. Damavandpeyma, S. Stuijk, T. Basten, M. Geilen, and H. Corporaal, "Throughput-constrained dvfs for scenario-aware dataflow graphs," in *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2013, pp. 175–184.

[16] A. K. Singh, A. Das, and A. Kumar, "Energy optimization by exploiting execution slacks in streaming applications on multiprocessor systems," in *Proceedings of the 50th Annual Design Automation Conference*, ser. DAC '13. New York, NY, USA: ACM, 2013, pp. 115:1–115:7. [Online]. Available: http://doi.acm.org/10.1145/2463209.2488875

[17] J. Zhu, I. Sander, and A. Jantsch, "Energy efficient streaming applications with guaranteed throughput on mpsocs," in *Proceedings of the 8th ACM International Conference on Embedded Software*, ser. EMSOFT '08. New York, NY, USA: ACM, 2008, pp. 119–128. [Online]. Available: http://doi.acm.org/10.1145/1450058.1450075

[18] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proceedings of the 2007 International Symposium on Low Power Electronics and Design*, ser. ISLPED '07. New York, NY, USA: ACM, 2007, pp. 38–43. [Online]. Available: http://doi.acm.org/10.1145/1283780.1283790

[19] T. Kolpe, A. Zhai, and S. S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[20] A. Colin, A. Kandhalu, and R. Rajkumar, "Energy-efficient allocation of real-time applications onto heterogeneous processors," in *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug 2014, pp. 1–10.

[21] D. Liu, J. Spasic, G. Chen, and T. Stefanov, "Energy-efficient mapping of real-time streaming applications on cluster heterogeneous MPSoCs," in *2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, Oct. 2015, pp. 1–10.

[22] J. Spasic, D. Liu, and T. Stefanov, "Exploiting Resource-constrained Parallelism in Hard Real-time Streaming Applications," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, ser. DATE '16. San Jose, CA, USA: EDA Consortium, 2016, pp. 954–959. [Online]. Available: http://dl.acm.org/citation.cfm?id=2971808.2972028

[23] M. Pelcat, K. Desnos, J. Heulot, C. Guy, J.-F. Nezan, and S. Aridhi, "Preesm: A dataflow-based rapid prototyping framework for simplifying multicore dsp programming," in *Education and Research Conference (EDERC), 2014 6th European Embedded Design in*, Sept 2014, pp. 36–40.

[24] M. Pelcat, J. F. Nezan, J. Piat, J. Croizer, and S. Aridhi, "A System-Level architecture model for rapid prototyping of heterogeneous multicore embedded systems," DASIP, 2009.

[25] H.Rexha, S.Holmbacka, and S.Lafond, "Core level utilization for achieving energy efficiency in heterogeneous systems," pp. 1–7, Mar 2017.

[26] O. J. Arndt, D. Becker, C. Banz, and H. Blume, "Parallel implementation of real-time semi-global matching on embedded multi-core architectures," in *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, July 2013, pp. 56–63.

[27] K. Desnos, M. Pelcat, J.-F. Nezan, and S. Aridhi, "Memory analysis and optimized allocation of dataflow applications on shared-memory mpsocs," *J. Signal Process. Syst.*, vol. 80, no. 1, pp. 19–37, Jul. 2015. [Online]. Available: http://dx.doi.org/10.1007/s11265-014-0952-6

[28] J. Lelli, C. Scordino, L. Abeni, and D. Faggioli, "Deadline scheduling in the Linux kernel," *Software: Practice and Experience*, vol. 46, no. 6, pp. 821–839, Jun. 2016. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/spe.2335/abstract

Paper 4

Energy Efficiency Platform Characterization for Heterogeneous Multicore Architectures

Hergys Rexha, Sébastien Lafond

Originally published *Proceedings of the 6th International Conference on ICT for Sustainability,ICT4S 2019, Lappeenranta, Finland, June 10-14, 2019*

4

Energy Efficiency Platform Characterization for Heterogeneous Multicore Architectures

Hergys Rexha
Faculty of Science and Engineering
Åbo Akademi University
Turku, Finland
hrexha@abo.fi

Sébastien Lafond
Faculty of Science and Engineering
Åbo Akademi University
Turku, Finland
slafond@abo.fi

***Abstract*—Runtime estimation of power dissipation and performance is crucial in every computing platform. In mobile systems, a special focus is set on energy efficiency in order to achieve the longest possible battery life and at the same time adhering to performance requirements. Powered by heterogeneous SoC's, mobile systems are called to reach an energy efficient state of execution, with a runtime system or scheduler that requires knowledge on the current performance and power dissipation. Today, highly heterogeneous architectures provide many actuators to reach better efficiency, the effect of which is usually unknown at runtime. In this paper, we propose a fast approach to build an energy efficiency model based on hardware performance counters. Our approach obviates the need for power sensors present at the chip level and deals with high numbers of execution modes. In building the energy efficiency model we account for the change in temperature which, as we show, has an impact on the optimal energy efficiency choice. The proposed approach reduces significantly the time to characterize the energy efficiency of a Multiprocessor System-on-Chip (MPSoC) and includes the environment temperature as a variable in determining the energy efficiency.**

***Index Terms*—MPSoC, energy efficiency models, platform configuration point, PMC, power models**

I. Introduction

The past years have seen rapid development in the amount of data produced, processed and exchanged through computing systems, ranging from high-end server farms to simple household devices, and the trend of technology seems to fuel even more this direction. Based on electricity usage ascribed to Information and Communication Technology (ICT), it is predicted that by the end of 2030 this sector will use as much as 51% of global electricity production [5]. Following this scenario, by the year 2030, the only ICT industry will be responsible for up to 23% of the globally released greenhouse gas emissions [5]. A 2016 report [24] says that the US datacenters held 350 million terabytes of data in 2015, and by 2020 they will require 100TWh of electricity to operate. This is the equivalent of 7 nuclear power stations like Olkiluoto 3 in Finland. There is also an increase of datacenters capacity in Europe, with London, Frankfurt, Paris, and Amsterdam which grew their electricity consumption by 200MW in 2017. Countries like Ireland and Denmark in Europe are becoming a data base for the world's biggest tech companies and by the next 5 years promise to increase the power consumption by 1TW [12]. The emergence of the Internet of Things (IoT) with devices operating at the edge of the network, poses a new challenge to the Cloud to provide efficient service provisioning. IoT devices are low powered devices and their usage promises to decrease the overall power consumption by increasing energy efficiency, but their number could be overwhelming with the consequence of having a "rebound effect" [9]. Cisco predicts that by the year 2020 in the world will be 50 billion IoT devices, which is an order of magnitude bigger than the number of smartphones and tablets working today. So in this scenario, using the cloud services offered by large datacenters to receive the data generated by IoT devices will not be a sustainable solution in terms of cost, latency, and environmental impact [6]. Recently the idea of edge devices that provide the computation and storage closer to the source of data has been formulated under the term of Edge or Fog computing [25]. As an edge device example, we can mention smartphones, as intermediates between body sensors and the cloud services, gateways as intermediates for smart homes, or nano data centers that manage the caching or processing of video contents. By using these edge devices in the proximity of data sources, we could have as an end result in a reduction of energy consumption w.r.t. implementing the logic in the cloud, and at the same time keeping latency requirements of certain applications [17].

Therefore one key requirement of such computing systems is undoubtedly energy efficiency. Basically, this means that systems should minimize their energy consumption to complete the required task and achieve a satisfying energy proportionality [20]. One of the largest consumers of energy in computing environments is the CPU [8], which requires special attention especially in the multicore era. Today mobile devices are using the same CPU as traditional gateways or cloudlets in Edge Computing. The need to achieve energy efficiency in today's MPSoC is stringent, especially for mobile devices that operate on battery, and that is a clear scenario where the end user wants a better experience and longer battery life.

Workload variability makes the control of energy expenditure especially difficult in mobile CPUs. Mobile devices are not the only which require energy efficient solutions, but also cloud providers need to lower the energy cost of

computations and cooling [19]. Today large scale computing facilities are using energy as a resource to be scheduled and charge according to the energy consumption [14]. Heterogeneity shows a promise to increase the energy efficiency levels achieved in MPSoC, hence several paths have been followed by research and industry. For example, exploring heterogeneity inside the CPU chip by using multiple technologies with different power and performance characteristics or using cores that alternatively behave as out-of-order computing elements or as in-order cores [22]. Probably one of the most popular and researched types of heterogeneity is the one provided by different computing cores integrated into the same physical chip. This type of heterogeneity is the one where computing cores share the same Instruction Set Architecture (ISA) but have different microarchitectures. However, an intelligent use of these power and performance tradeoffs proves to be not a simple challenge [23]. Being able to predict the optimal choice between a number of hardware actuators such as the number of cores, type of core and operating performance point, or Dynamic Voltage and Frequency Scaling (DVFS), is a difficult task that must be handled well in order to achieve energy efficiency.

With asymmetric multiprocessing (AMP) architecture there is a better way to respond to the diversity of applications present in the mobile environment. We have compute-intensive applications which need to produce results in real time and must use fast cores in order to meet the deadlines. On the other side, background processes that may be memory bound require little computation and are more suitable to run on simple cores that achieve better levels of energy efficiency. Even within a single application, we have different "windows of activity" which may require varying levels of computing intensity, e.g. reading, scrolling, responding through different messages inside a social media application. Recently industry has moved towards increasing the level of heterogeneity found inside a single chip. From examples such as ARM big.LITTLE with two types of cores, to Mediatek tri-cluster MPSoC [16] which promise to increase performance and reduce power dissipation. DynamIQ from ARM [1] advances the concept of big.LITTLE by providing better flexibility in the cluster organization and frequency setting.

High levels of heterogeneity present in recently embedded architectures produce an increase in the design space exploration to find an efficient use of platform actuators. By increasing the number and type of cores and the number of voltages and frequency levels for each computing element, there is an increasing number of operating points on which the platform may perform. In this scenario making the right choice for execution could have a tremendous impact on energy efficiency. Temperature also has a major effect on the power dissipation of today's systems [15], which makes it an important factor to account for in order to make the optimal energy efficient choice.

To manage efficiently the workload scenarios faced by mobile devices, edge devices in IoT, or nano data centers, there is a need to continuously monitor power data in order to choose the optimal power and performance trade-off. Unfortunately, most of the hardware platforms today are not equipped with power sensors, which significantly complicates energy-efficient management of the system settings.

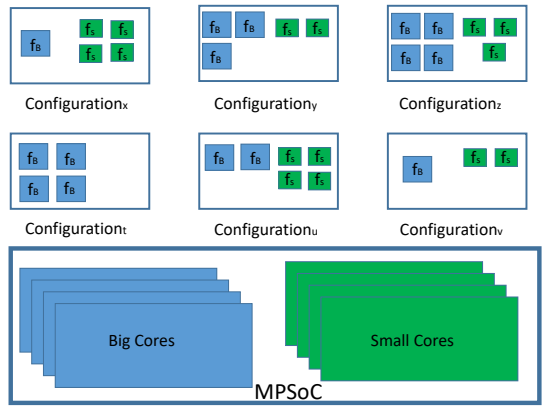

Fig. 1. Examples of possible platform configuration points in a multicore architecture

This paper follows our previous work which experimentally builds an energy efficiency model based on platform configuration points, for ARM big.LITTLE architecture [21]. As platform configuration point we denoted the set of platform actuators such as *number, type of core, core performance level or DVFS and core utilization level*. The model is derived by testing all the possible configuration points of the platform. Following the recent trend in platform complexity, this approach is difficult to apply in the case of the combinatorial explosion in the number of configuration points. The goal of this paper is to explore new approaches in providing knowledge of the platform energy efficiency to a runtime system based on the concept of platform configuration points. We redefine the set of parameters in the configuration point by removing *utilization level* from the aforementioned description. Meaning of the notion of platform configuration point is demonstrated with several examples (from x to v) in a multicore platform (Figure 1). In our energy efficiency model, we account for the environment temperature variable, which provides valuable information for the correct accounting of the CPU dissipated power. Knowing the large impact that static power has on the energy efficiency achieved in today's CPUs the second purpose of this work is to build thermally aware energy efficiency models.

The contributions of this paper are the following:

- we propose an approach to characterize the energy efficiency of a hardware platform based on the notion of configuration points.
- we include environment temperature in the energy efficiency model and show the impact this variable has on the relative efficiency of the points from the model.

II. Related Work

Exploring the usage of platform actuators for energy management was studied by different research works. The authors in [23], [10], and [18] all propose the creation of a runtime system which is able to manage the scheduling and mapping of threads dynamically with the objective of maximizing the energy efficiency of MPSoC. In [23] a load balancer schedules the workload in periodic time frames called *epochs*, wherein each, a set of actions are performed to set the threads in the appropriate core type. The platform considered is highly heterogeneous with 4 types of core and in each *epoch* the load balancer estimates the performance and power of every thread in each core type. This information is used by the internal algorithm to decide where to map the threads. Similarly, in [18] is proposed a runtime scheme which is used to schedule dynamically workloads in a MPSoC. The approach is based on the sense-decide-act policy and operates on an aggressive heterogeneous environment. It uses regression models for estimating performance and power of threads in different core type and also the contribution of a thread in a total load of a core. An evolutionary algorithm is used to decide in each term the scheduling of the threads. The authors in [10] propose a run-time task allocation approach called SPARTA which categorizes task in computing bound or memory bound and a heuristic that selects the configuration that achieves the requested throughput with the minimal power consumption. In these works is not considered the possibility of DVFS as a mechanism to reduce power consumption and also the hardware counters used for estimating performance are not easily found in real hardware platforms. Sensors for estimating the power consumption of different mapping decisions are not available in many of today's platforms. Finding the optimal configuration for executing workloads in a data-center in order to achieve better energy efficiency is the goal presented in [11]. Authors present a programming and execution platform called Empya that uses hardware and software techniques to determine the best trade-off between performance and energy consumption. The run-time system continuously monitors application performance and energy consumption through Running Average Power Limit (RAPL) registers. As actuators, the system operates on the number of threads to use and the power cap on the CPU. In contrast with this, our work focuses on heterogeneous platforms where for achieving energy efficiency we use actuators such as number, type of core and DVFS point. In [26] authors target again High-Performance Computing applications running on a single node with the goal of reducing the energy consumption by choosing the right configuration, which is composed of the number of cores and DVFS level. The work is based on the application-agnostic power model and the performance model of the application is obtained with a supervised learning method of regression. Frequency, number of cores and input size are used in the regression model. The methodology is clear and straightforward, but there is no mention of the performance requirement which is the value we trade off for less energy consumption.

III. CMOS power dissipation

CMOS technology has been mostly used in MPSoCs due to the fact that has quite good noise immunity and low heat production while the device is in operation mode. Power in these circuits can be divided into two categories: dynamic power and static power. Dynamic power is created by the circuit activity (transistor switching) and is dependent on the usage scenario, clock rates, and I/O activity. Switching power is dissipated during the transistor changing from 0 to 1 and vice versa, the dynamic power is defined as:

$$P_{\text{dynamic}} = \alpha * C * V_{DD}^2 * f_{clk} \tag{1}$$

where C is the load capacitance, V_{DD} is the source voltage, α is the activity factor and f is the operating frequency. Static power is dissipated due to the leakage currents on the transistors while they are in the "OFF" mode. The are several sources of the leakage current which are strongly influenced by the chip temperature. The dynamic part of the power dissipated from the chip is modeled by two terms in Equation 2, as a dynamic activity which relates to the active running workloads and the background activity that represents the system processes that run on the background. In Equation 3 the dynamic power is modeled by a single term due to the low power dissipated by background processes in the A7 cluster. Static power is modeled by the third term in Equation 2 and is dependent on temperature and the supply voltage. For the A7 cluster, there is no temperature sensor to monitor, hence the static part is modeled together with the dynamic power dissipation of background activity.

IV. Proposed Approach

Today embedded systems face a multitude of working scenarios that range from burst in high performance requests, to low power operation modes, going through the need to provide sustainable performance in thermally constrained situations. To do an efficient managing of such a number of use cases the runtime scheduling manager need to have refreshed information about the effect of changing different actuators on the running applications. Thus there is a need for an energy efficiency model which is based on the current runtime power data. The envisioned system diagram is shown in Figure 2, where our work in this paper is focused in providing the platform configuration points database for helping the scheduler decisions in reaching the optimal efficiency level of the running applications.

The work in this paper is based on power models for mobile CPUs based on hardware program counters (HPC). The methodology for building such models is adopted from [27], which presents a statistical method for identifying and using hardware counters. Their analyses propose the usage of counters which show a high correlation to power and have also the

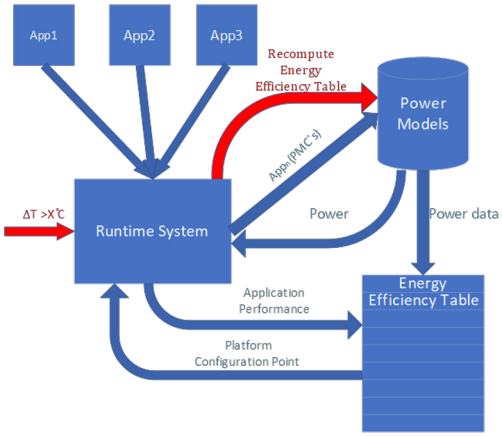

Fig. 2. Proposed Approach schematics.

TABLE I
HARDWARE EVENTS USED IN THE POWER MODELS

Event list		
Nr	**ARM Cortex-A7**	**ARM Cortex-A15**
1	L2D_CACHE_ACCESS:0x16	L2D_CACHE_LD:0x50
2	MEM_ACCESS:0x13	DP_SPEC:0x73
3	L1I_CACHE_ACCESS:0x14	L1I_CACHE_ACCESS:0x14
4	UNALIGNED_LDST:0x0F	UNALIGNED_LDST_SP:0x6A
5	CYCLE_COUNT:0x11	BUS_ACCESS:0x19
6		INST_SPEC:0x1B
7		CYCLE_COUNT:0x11

smallest multicollinearity. The authors in [27] show that this brings high model stability with an average error of 3,8%.

We start by building power models for two popular ARM v7a architecture CPU's, which are ARM Cortex-A7 and ARM Cortex-A15. The micro-architecture limits the number of events which can be sampled at once: 6 counters for A15 and 4 counters for A7 plus the cycle counter. The goal is to search for those events which have the highest correlation with power dissipation and at the same time show the smallest intercorrelation with each other. To have high model stability the predictors should be chosen to keep low levels of multicollinearity in multivariate models. First, is measured the correlation of all available events with the power, then the counters are divided into clusters which include events with high intercorrelation. Then, from each cluster is selected the event which has more impact on the power dissipation but keeping a low Variance Inflation Factor (VIF). The total amount of events for the A7 is 40 and for the A15 in 120, among these are selected 7 for the A15 and 5 for the A7. The events used in the models are general and can be found on most core types used in mobile systems. For each core type, the events are listed on Table I. The power for A15 and A7 is divided in dynamic and static, plus the background power which is related to the operating system activities. The modelled formula for the power dissipation is showed in Equation 2 and 3,

$$P_{\mathrm{A15}} = \underbrace{(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk})}_{\text{dynamic activity}} + \underbrace{\beta_b V_{DD}^2 f_{clk}}_{\text{BG dynamic}} + \underbrace{f(V_{DD}, T)}_{\text{static}} \tag{2}$$

$$P_{\mathrm{A7}} = \underbrace{(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk})}_{\text{dynamic activity}} + \underbrace{f(V_{DD}, f_{clk})}_{\text{static and BG dynamic}} \tag{3}$$

where N is the number of events selected, β_n is the weight given to certain event, E_n is the number of events per second divided by the frequency (f_{clk}) in MHz, V_{DD} is the operating voltage and T is the temperature of the core.

The power model for the A15 has a thermal compensation term for calculating the static power and background dissipated power when the system is idling (Equation 2). In the power model for A7 the static and background power are included in the second term of Equation 3. This is related to the absence of a thermal monitoring sensor in the A7 cluster. We have calculated four sets of model coefficients for the parameters in each cluster, representing the power with a different number of cores for each CPU type. The model parameters for each core type are given in Tables II and III. In the tables, it is shown the event rate divided by the frequency in MHz, the weight given to each coefficient and the statistical significance. In some model terms, f and V are respectively the operating frequency and voltage of each cluster (Table IV). The event rates are divided by the operating frequency in order to avoid correlation with it in the first term of power equations. The power models need to be obtained only once by running on the target platform a set of embedded representative workloads which we call platform characterization set. After obtaining the power model we compute the energy efficiency table which provides a sort of database of all the possible platform configuration points and the resulting performance, power and energy efficiency values. By having this information the runtime system is able to make decisions about the mapping of a certain application with regard of the performance. If there is a change in the environment temperature above a certain threshold, then the power dissipation can be recomputed and the table is redefined for the new thermal level.

These models are build by running the characterization workload set in each of the operating points of both CPUs. The set contains workloads that test different levels of the microarchitecture and memory subsystem. In part is composed of real applications from the embedded domain, and for the other part synthetic benchmarks designed to stress specific parts of the CPU. Having the power models and by measuring the performance in terms on instructions per second (IPS) we can obtain an energy efficiency model of the platform. The model is presented as a table that lists all the platform configuration points with the energy efficiency levels achieved in

TABLE II
MODEL PARAMETERS AND P-VALUES FOR THE A15

Nr	Coefficient	Weight	p-Value
1	Intercept	-5e-4	p<e-4
2	$EPH\_0x11 * f * V^2$	7.9e-10	p<e-4
3	$(EPH\_0x1b - EPH\_0x73) * f * V^2$	e-10	p<e-4
4	$EPH\_0x50 * f * V^2$	8.7e-9	p<e-4
5	$EPH\_0x6a * f * V^2$	e-8	p<e-4
6	$EPH\_0x73 * f * V^2$	2.6e-11	p<2e-3
7	$EPH\_0x14 * f * V^2$	6.4e-11	p<e-3
8	$EPH\_0x19 * f * V^2$	1.9e-9	p<e-4
9	V	0.17	p<e-4
10	$f * V^2$	1.6e-4	p<e-4
11	T	2.3e-2	p<e-3
12	T^2	2.9e-4	p<4e-3
13	$V * T^2$	-3.5e-5	p<e-3
14	$V * T$	1.1e-2	p<e-3

TABLE III
MODEL PARAMETERS AND P-VALUES FOR THE A7

Nr	Coefficient	Weight	p-Value
1	Intercept	-7.2e-4	p<0.003
2	$EPH\_0x11 * f * V^2$	1.9e-10	p<e-4
3	$EPH\_0x14 * f * V^2$	2.2e-10	p<e-4
4	$EPH\_0x13 * f * V^2$	4.3e-10	p<e-4
5	$EPH\_0x16 * f * V^2$	1.4e-9	p<e-4
6	$EPH\_0x0f * f * V^2$	9.4e-11	p<0.0004

terms of instructions per Joule, performance point (instructions per second) and the power dissipation (W). The table is used to decide the optimal configuration point for an application that has defined performance requirements. Once an application is submitted into the system or is resumed by the scheduler, the runtime system can sample the hardware counters in a single frequency level and scans the table to find the optimal configuration point, to run the application, in terms of energy efficiency. In this work, we consider multi-threaded applications, which matches our methodology of achieving optimal levels of energy efficiency by using configuration points that possibly use several cores. In the case where the performance requirement of the application changes, the control logic of the runtime system can select another configuration point that provides the requested performance level and has a high level of energy efficiency. When the temperature of the environment changes above a certain threshold, the power model can be used to recompute the energy efficiency table in accordance with the new temperature conditions. A temperature increase in the outside environment produces an increased level of static power in the CPU, which affects the relative efficiencies of the configurations inside the energy efficiency table. The runtime system can continuously monitor the power usage of the running application in order to not exceed the Thermal Design Power (TDP) of the CPU. By sampling the performance counters of each running application the power model shows the power dissipation at runtime of the running applications, thus the runtime system can make a decision of reducing the power dissipation of certain applications by choosing another configuration point from the system.

The runtime system inputs temperature variations inside the model and can recompute the energy efficiency table by taking into account the new level of static power. The new table needs to be searched for configuration points that satisfy the performance request with the highest level of efficiency. A basic schematic of the proposed approach is given in Figure 2.

V. EXPERIMENTAL SETUP

To evaluate our approach we used an ODROID XU3 development board from HARDKERNEL. The application processor implements the ARM big.LITTLE architecture with two clusters composed of 4 cores each. The big cluster consists of a high-performance Cortex-A15 quad-core block, and a low power Cortex-A7 quad-core CPU. The board description is complete with a Mali-T628 GPU and 2GB LPDDR3 of memory. The board contains 4 current sensors that offer the possibility to measure power dissipation in four different domains: big cluster (A15), LITTLE cluster (A7), GPU and memory. Besides this, the board contains 4 temperature sensors for the cores in the big cluster and one temperature sensor for the GPU. The characteristics of the hardware can be found in Table IV.

TABLE IV
CHARACTERISTICS OF THE EXPERIMENTAL BOARD

Characteristic	ODROID Development Board
Model	XU3
SoC	Exynos 5422 Octa core
CPU's	Cortex-A15/A7
cores	4 + 4
Frequency A7 (MHz)	
min	200
max	1400
Frequency A15 (MHz)	
min	200
max	2000
Voltage A7 (V)	
min	0.9
max	1.24
Voltage A15 (V)	
min	0.9
max	1.36

To build the power model we used a set of benchmarks from different application domains. We call the training set as the platform characterization workloads. In the platform characterization set we include a sequence of 76 workloads which consists of a collection of synthetic and real world applications from Roy Longbottom [4], PARSEC [7], CoremarkPro [2], ParMiBench [13] and Multibench [3]. A full list of the used workloads is in Table V.

The choice of the workload set is based on the idea of all-inclusiveness of applications that characterize the embedded systems domain.

Experiments were conducted in different environments to account for the outside temperature change in the SoC power dissipation. The goal here is to evaluate the change in the energy efficiency table in accordance with temperature. For the first environment, the board fan is running with 100% speed with the system located in a highly refrigerated environment.

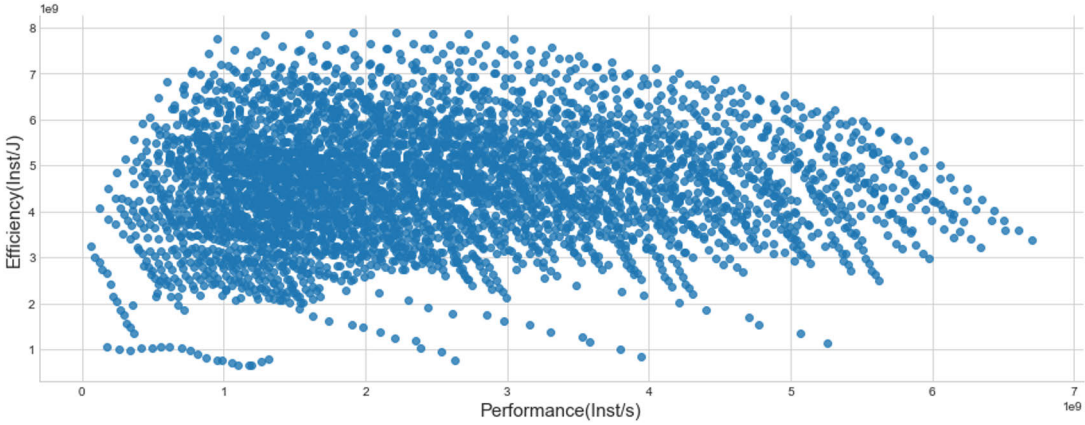

Fig. 3. Configuration points from the model

In the second case, the board is working with the fan disabled in a normal outside temperature to account for a high temperature outside the environment. In the third case, the board is working with the fan always on in a normal environment, to justify the middle case. In Table, VI on Section V we will show the result of the energy efficiency table computed in different environments.

VI. Results

By using the power and performance models defined previously we are able to derive an energy efficiency model which is based on platform configuration points. In Figure 3 we show the efficiency of all configuration points from the model. Each point describes a single configuration that provides a certain level of performance in terms of instructions per second and energy efficiency. By going towards high levels of performance we notice a decrease in the density of the points. This means that fewer options for achieving good energy efficiency levels. The list of configurations is organized as an energy efficiency table that lists all possible configuration points with their efficiency and performance. An example of the table derived from the workloads in the training set of the power model is shown in Table VII. By searching inside the table we find several sets of configuration points that provide the same performance but with different energy efficiency levels, some of the sets are shown in Figure 4. First usage of the table would be the one for choosing the optimal configuration point based on a certain requirement for the performance level. As it is shown by Figure 4, it is possible to gain in terms of energy efficiency if we make the right choice for the configuration point. As a second objective of our work, we wanted to test the effect of temperature on the relative energy efficiency of configuration points in the model. For testing thermal effects on the efficiency model, we choose to run a testing application with the system located in different environments. We run Basicmath application from the ParMiBench suite [13]. In environment 1, the system running in a highly refrigerated environment (we call it "cold" case). In Environment 2, the system is running without a fan with an outside temperature of 25°C (we call it "hot" case). Environment 3, consists of the system running on a 25°C outside temperature with the fan always on at 100% speed (we call it "middle" case). We noticed the relative order of configuration points changes between the environments and so does the energy efficiency levels achieved.

The top rows of the energy efficiency table for different temperature environments are shown in Table VI. Different temperature levels produce different order of configuration points and efficiency levels achieved. This shows that there is a need to change the platform configuration point when the temperature changes significantly, in order to keep the high levels of energy efficiency.

In Figure 5 we show a possible runtime scenario. We are running Basicmath test application with a required level of performance such as e.g. 1,61E+9 inst/s in a system with a temperature t_1, according to the model the optimal configuration point for this performance level is composed by 2a7@400MHz + 4a15@500MHz. In the case, the temperature increases to t_2, then the efficiency of that configuration point decreases and thus we need to reconfigure with the new table that shows that we should execute the application by using the following configuration 4a7@700MHz + 4a15@200MHz. Another example is shown with the performance requirement of 3,27E+9 inst/s, where again there is a need for reconfiguration in order to keep high levels of energy efficiency.

The change in the environment temperature of the system (from "cold" to "hot") produces large differences in the energy efficiency levels that the model defines as an optimal configuration point for the required performance. By looking at the first 100 highly energy efficient configurations in the energy efficiency table, we find few test cases, whereby changing the configuration point when the system temperature changes the

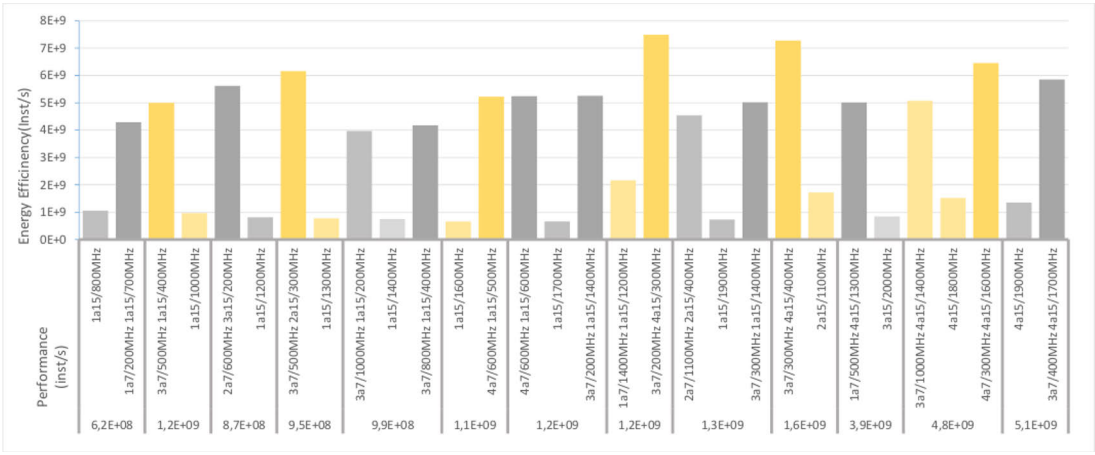

Fig. 4. List of configuration points grouped in different performance classes

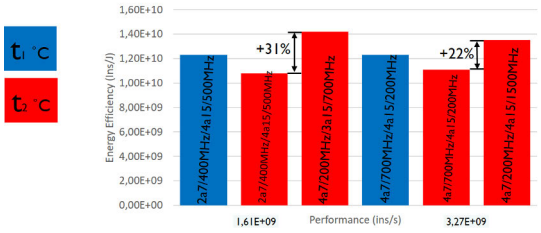

Fig. 5. Reconfigure examples in two temperature environments

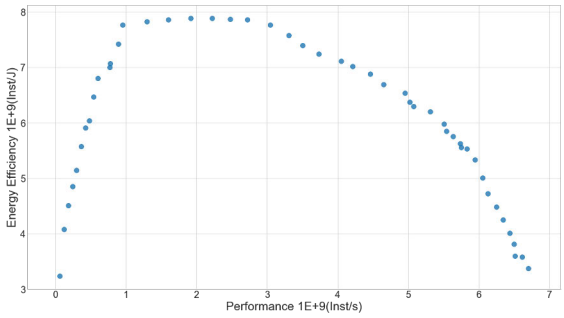

Fig. 6. Configuration points with high energy efficiency levels

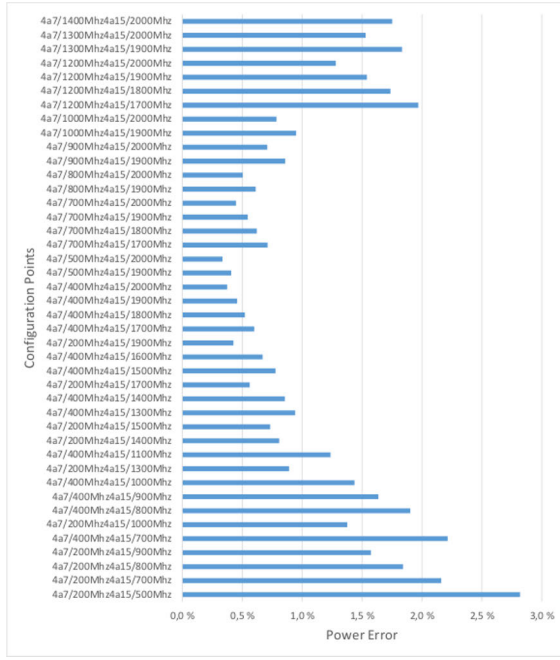

Fig. 7. Power errors for configuration points with high level of energy efficiency

gain in terms of energy efficiency is up to 33%. By searching for new target reconfiguration points we account for the same performance or 5% bigger. An interesting observation can be noticed in Figure 3 where all points are plotted in the energy efficiency and performance graph. If we take the points from the upper outer layer of the scatter plot we have a situation like in Figure 6. Those points show the configurations with the optimal energy efficiency for a certain level of performance at a defined temperature. Or otherwise, we can think of the graph as the result of scanning the model from the lowest

TABLE V
PLATFORM CHARACTERIZATION SET

List of benchmarks	
Suite	Workload
CoremarkPro	core linear_alg-mid-100x100-sp loops-all-mid-10k-sp nnet_test parser-125k radix2-big-64k sha-test zip-test
MultiBench	4M-check 4M-check-reassembly 4M-check-reassembly-tcp 4M-check-reassembly-tcp-cmykw2-rotatew2 4M-check-reassembly-tcp-x264w2 4M-cmykw2 4M-cmykw2-rotatew2 4M-reassembly 4M-rotatew2 4M-tcp-mixed 4M-x264w2 empty-wld iDCT-4M iDCT-4Mw1 ippktcheck-4M ippktcheck-4Mw1 ipres-4M ipres-4Mw1 md5-4M md5-4Mw1 rgbcmyk-4M rgbcmyk-4Mw1 rotate-4Ms1 rotate-4Ms1w1 rotate-4Ms64 rotate-4Ms64w1 x264-4Mq x264-4Mqw1
MiBench	automotive/qsort network/dijkstra consumer/typeset telecomm/adpcm
Parsec-3.0	blackscholes bodytrack canneal dedup ferret fluidanimate freqmine streamcluster swaptions
ParmiBench	Office/stringsearch Network/Patricia/Parallel Automotive/Susan/Parallel Automotive/Bitcount/Parallel Network/Dijkstra/Parallel Office/stringsearch/Parallel
Roy-Longbottom	rl-linpack-neon rl-linpack-FSSP rl-whetstone rl-busspeed rl-dhrystone
Lmbench	lat_ctx lat_fs lat_ops lat_proc lat_fifo lat_http lat_pagefault lat_select lat_sem lat_unix_connect lat_mem_rd bw_mem tlb lmb3-tlb line
Whetstone	whetstone
Drystone	dhrystone

TABLE VI
TOP ENERGY EFFICIENCY CONFIGURATIONS FOR THREE ENVIRONMENTS

Temperature Environment 1			
Configuration	***Energy Efficiency (Ins/J)***	***Power(W)***	***Performance(Ins/s)***
4a7/200MHz4a15/500MHz	1,517e+10	0,465	1,61e+09
4a7/200MHz4a15/700MHz	1,515e+10	0,599	2e+09
4a7/200MHz4a15/400MHz	1,512e+10	0,382	1,39e+09
4a7/200MHz4a15/300MHz	1,511e+10	0,305	1,17e+09
4a7/200MHz4a15/200MHz	1,50e+10	0,219	9,37e+08
....	....	....	....
Temperature Environment 2			
4a7/200MHz3a15/300MHz	1,424e+10	0,333	9,92e+08
4a7/200MHz3a15/500MHz	1,421e+10	0,518	1,32e+09
4a7/200MHz3a15/400MHz	1,420e+10	0,428	1,15e+09
4a7/200MHz3a15/600MHz	1,420e+10	0,608	1,48e+09
4a7/200MHz3a15/700MHz	1,416e+10	0,697	1,61e+09
....	....	....	....
Temperature Environment 3			
4a7/200MHz4a15/600MHz	1,49e+10	0,586	1,82e+09
4a7/200MHz4a15/400MHz	1,49e+10	0,415	1,39e+09
4a7/200MHz3a15/700MHz	1,49e+10	0,668	2e+09
4a7/200MHz3a15/500MHz	1,480e+10	0,511	1,61e+09
4a7/200MHz3a15/300MHz	1,486e+10	0,337	1,17e+09
....	....	....	....

TABLE VII
ORDERED ENERGY EFFICIENCY TABLE .

C	C($N_l/F_l/N_b/F_b$)	Perf.(inst/s)	P_{avg}(W)	Efficiency(inst/J)
1	4a7/200MHz/4a15/600MHz	2.219115e+09	0.699744	7.889801e+09
2	4a7/200MHz/4a15/500MHz	1.916094e+09	0.600826	7.885497e+09
3	4a7/200MHz/4a15/700MHz	2.475814e+09	0.788427	7.872383e+09
4	4a7/200MHz/4a15/800MHz	2.723064e+09	0.873142	7.861730e+09
5	4a7/200MHz/4a15/400MHz	1.601398e+09	0.501352	7.857119e+09
6	4a7/200MHz/4a15/300MHz	1.294310e+09	0.402370	7.830159e+09
7	4a7/200MHz/4a15/900MHz	3.042998e+09	1.010040	7.765476e+09
8	4a7/200MHz/4a15/200MHz	9.541939e+08	0.293673	7.763320e+09
9	4a7/300Mhz 4a15/600MHz	2.338974e+09	0.728441	7.647120e+09
10	4a7/300Mhz 4a15/500MHz	2.035953e+09	0.629523	7.642816e+09
11	4a7/300Mhz 4a15/700MHz	2.595672e+09	0.817124	7.629703e+09
12	4a7/300Mhz 4a15/800MHz	2.842923e+09	0.901839	7.619049e+09
13	4a7/300Mhz 4a15/400MHz	1.721256e+09	0.530049	7.614439e+09
14	4a7/300Mhz 4a15/300MHz	1.414169e+09	0.431067	7.587478e+09
15	4a7/200Mhz 4a15/1000MHz	3.310742e+09	1.173238	7.580142e+09
.	.	.	.	.
4078	1a15/1800MHz	1.193975e+09	1.795146	6.651129e+08
4079	1a15/1700MHz	1.176482e+09	1.776230	6.623477e+08
4080	1a15/1600MHz	1.101565e+09	1.670471	6.594337e+08

performance point and keeping only those points which have higher performance and the highest possible level of energy efficiency. As a further validation of our approach, we measure in percentage the difference between the predicted power dissipation and the measured power in configuration points with high levels of energy efficiency. The results are shown in Figure 7, where we notice the highest error is 2,82%. We measure the model errors in configurations that provide the highest levels of energy efficiency for different performance levels. These are more intriguing configuration points, which give the best of the platform's energy efficiency. Knowing that most of the time these points will be used as configuration options, having a low error rate from the model is very useful.

VII. CONCLUSION

In this work, we present an approach for building an energy efficiency model which is based on platform configuration points. The target of the approach are heterogeneous platforms which are continuously increasing the depth of heterogeneity. The model is based on hardware performance counters which are widely available in today's CPU architectures. The set of workloads for building the model is representative of the

embedded domain which has shown to be more critical to the energy efficient application execution. But also, the training set, in inclusive of the IoT world. The novelty of this approach compared to previous works is that it doesn't necessarily need power sensors for measuring the power dissipation in each configuration point, but by sampling the counters on one configuration point we can characterize the efficiency of other configuration points. From all the points in the model, we show that less than 1% of them (see points in Figure 7) represent the highest levels of energy efficiency possible, in all the performance spectrum offered by the platform. Also, we include the environment temperature as a variable for defining the need for application reconfiguration. As we show by the tests if the temperature changes, by reconfiguring the application execution we can gain up to 33% in terms of energy efficiency.

REFERENCES

[1] Arm DynamIQ Technology for the next era of compute - Processors blog - Processors - Arm Community. https://community.arm.com/processors/b/blog/posts/arm-dynamiq-technology-for-the-next-era-of-compute.

[2] CPU Benchmark – CoreMark-PRO – EEMBC Embedded Microprocessor Benchmark Consortium. https://www.eembc.org/coremark-pro/index.php.

[3] EEMBC - MultiBench - Multicore Benchmark. https://www.eembc.org/multibench/.

[4] Roy Longbottom's PC Benchmark Collection - Free PC Benchmarks. http://www.roylongbottom.org.uk/.

[5] Anders S. G. Andrae and Tomas Edler. On global electricity usage of communication technology: Trends to 2030. *Challenges*, 6(1):117–157, 2015.

[6] M. Ashouri, P. Davidsson, and R. Spalazzese. Cloud, edge, or both? towards decision support for designing iot applications. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 155–162, Oct 2018.

[7] Christian Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[8] W. L. Bircher and L. K. John. Complete System Power Estimation Using Processor Performance Events. *IEEE Transactions on Computers*, 61(4):563–577, April 2012.

[9] Peter M. Corcoran. Third time is the charm - why the world just might be ready for the internet of things this time around. *CoRR*, abs/1704.00384, 2017.

[10] Bryan Donyanavard, Tiago Mück, Santanu Sarma, and Nikil Dutt. SPARTA: Runtime Task Allocation for Energy Efficient Heterogeneous Many-cores. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES '16, pages 27:1–27:10, New York, NY, USA, 2016. ACM.

[11] C. Eibel, T. Do, R. Meissner, and T. Distler. Empya: Saving Energy in the Face of Varying Workloads. In *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pages 134–140, April 2018.

[12] EIRGRID. All-island generation capacity statement 2017-2026, 2017.

[13] S. M. Z. Iqbal, Y. Liang, and H. Grahn. ParMiBench - An Open-Source Benchmark for Embedded Multiprocessor Systems. *IEEE Computer Architecture Letters*, 9(2):45–48, February 2010.

[14] V. Jimenez, F. Cazorla, R. Gioiosa, E. Kursun, C. Isci, A. Buyuktosunoglu, P. Bose, and M. Valero. Energy-Aware Accounting and Billing in Large-Scale Computing Facilities. *IEEE Micro*, 31(3):60–71, May 2011.

[15] J. S. Lee, K. Skadron, and S. W. Chung. Predictive Temperature-Aware DVFS. *IEEE Transactions on Computers*, 59(1):127–133, January 2010.

[16] H. Mair, E. Wang, A. Wang, P. Kao, Y. Tsai, S. Gururajarao, R. Lagerquist, J. Son, G. Gammie, G. Lin, A. Thippana, K. Li, M. Rahman, W. Kuo, D. Yen, Y. Zhuang, U. Fu, H. Wang, M. Peng, C. Wu, T. Dosluoglu, A. Gelman, D. Dia, G. Gurumurthy, T. Hsieh, W. Lin, R. Tzeng, J. Wu, C. Wang, and U. Ko. 3.4 A 10nm FinFET 2.8GHz tri-gear deca-core CPU complex with optimized power-delivery network for mobile SoC performance. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 56–57, February 2017.

[17] Jozef Mocnej, Martin Miškuf, Peter Papcun, and Iveta Zolotová. Impact of Edge Computing Paradigm on Energy Consumption in IoT. *IFAC-PapersOnLine*, 51(6):162–167, 2018. 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018 Citation Key: MOCNEJ2018162.

[18] Tiago Mück, Santanu Sarma, and Nikil Dutt. Run-DMC: Runtime Dynamic Heterogeneous Multicore Performance and Power Estimation for Energy Efficiency. In *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*, CODES '15, pages 173–182, Piscataway, NJ, USA, 2015. IEEE Press.

[19] V. Petrucci, M. A. Laurenzano, J. Doherty, Y. Zhang, D. Mossé, J. Mars, and L. Tang. Octopus-Man: QoS-driven task management for heterogeneous multicores in warehouse-scale computers. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 246–258, February 2015.

[20] L. Ramapantulu, D. Loghin, and Y. M. Teo. On Energy Proportionality and Time-Energy Performance of Heterogeneous Clusters. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 221–230, September 2016.

[21] Hergys Rexha, Simon Holmbacka, and Sebastien Lafond. Core Level Utilization for Achieving Energy Efficiency in Heterogeneous Systems. In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 401–407, St. Petersburg, Russia, 2017. IEEE.

[22] V. Saripalli, G. Sun, A. Mishra, Y. Xie, S. Datta, and V. Narayanan. Exploiting Heterogeneity for Energy Efficiency in Chip Multiprocessors. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(2):109–119, June 2011.

[23] Santanu Sarma, T. Muck, Luis A. D. Bathen, N. Dutt, and A. Nicolau. SmartBalance: A Sensing-driven Linux Load Balancer for Energy Efficiency of Heterogeneous MPSoCs. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 109:1–109:6, New York, NY, USA, 2015. ACM.

[24] Arman Shehabi, Sarah Josephine Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Nathaniel Horner, Inês Lima Azevedo, and William Lintner. United states data center energy usage report. Technical report, 06/2016 2016.

[25] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, Oct 2016.

[26] Vitor R. G. Silva, Alex Furtunato, Kyriakos Georgiou, Kerstin Eder, and Samuel Xavier-de-Souza. Energy-Optimal Configurations for Single-Node HPC Applications. *arXiv:1805.00998 [cs]*, May 2018.

[27] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett. Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(1):106–119, January 2017.

Paper 5

Towards Very Low-Power Mobile Terminals through Optimized Computational Offloading

Hergys Rexha, Sébastien Lafond, Giovanni Rigazzi, Jani-Pekka Kainulainen

Originally published *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*

Towards Very Low-Power Mobile Terminals through Optimized Computational Offloading

Hergys Rexha*, Sébastien Lafond*, Giovanni Rigazzi†, Jani-Pekka Kainulainen‡
*Åbo Akademi University, Faculty of Science and Engineering, Turku, Finland
‡InterDigital Europe, London, UK
†InterDigital Germany GmbH, Berlin, Germany

***Abstract*—Energy consumption is a major issue for modern embedded mobile computing platforms, and with new technological developments, such as IoT and Edge/Fog computing, the number of connected embedded mobile computing systems is rapidly increasing. Heterogeneous multi-core CPUs seek to improve the performance of these platforms, with a particular focus on energy efficiency. By using different techniques like DVFS, core mapping, and multi-threading, a substantial improvement in the achievable CPU energy efficiency level for Multi-processor system-on-chip (MPSoC) can be observed. However, controlling only the CPU power dissipation has a limited effect on the overall platform energy consumption. Other components of the platform, including memory, disk, and other peripherals, play an important role in the energy efficiency of the platform and need to be taken into account. The availability of different sleep strategies at various levels of the platform makes the energy efficiency issue even more complex. In this paper, we set the view of energy efficiency at the entire platform level and discuss computation offloading as a mechanism to help in reaching the optimal platform energy-efficient state. As an application, we consider object detection performed on several types of images to define when offloading is beneficial to the platform energy efficiency. We survey the energy efficiency of different neural network algorithms in an embedded environment, with the possibility to perform computation offloading, and discuss the obtained results concerning the level of object recognition accuracy provided by different neural networks.**

***Index Terms*—Energy efficiency, computation offloading, object recognition, embedded computing platforms.**

I. Introduction

In recent years, the number of connected devices has been dramatically increasing, with predictions for 2025 suggesting the number to reach over 8 Billion mobile broadband connections and over 5 Billion IoT connections [1]. Furthermore, new classes of services are emerging requiring support in future networks, such as rich 4K/8K video services for Mixed Reality (MR) applications with tactile feedback [1]. These, together with industrial automation control, autonomous ground, and air vehicles will provide further challenging requirements for future networks, noticeably from bandwidth, latency, reliability, and energy efficiency perspectives. Edge computing can theoretically provide high bandwidth, low latency, and the computing agility required by today's new digital services. On the other hand, Information and Communication Technology (ICT) drives an unstoppable process of ever-growing energy consumption, with new devices, services, and data produced at a rapid pace. The 5G and beyond network is set to support huge volumes (in the order of TBs of data per day [2]) of multi-dimensional data coming from different heterogeneous nodes, devices, and applications. This raises major challenges with respect to data transmission and processing. Ultimately, such a vast amount of data together with low latency application requirements, e.g., cloud gaming [3] calls for new approaches to perform processing in an energy-efficient manner. One key aspect of edge is that many of the devices are battery-powered or deployments are power limited. The more energy-efficient the edge processing is, the more computation can be done with the same power budget. Application complexity and power consumption are increased when distributed AI is deployed to edge devices. Furthermore, due to the ever-increasing number of such devices, the technology is expected to consume a significant amount of energy [4], thus playing a major concern in future strategies of curbing down energy consumption.

Multi-core heterogeneous CPUs promise to give a substantial contribution to the increase of the energy efficiency in edge platforms. Different strategies inside the MPSoC are explored to achieve a reduction in power dissipation and relative energy consumption. Nevertheless, it is not enough to consider only one component, e.g., at the CPU level, in the strategy for achieving energy efficiency, rather one must consider energy consumption at the platform level [5]. Depending on the platform type, I/O connected, and the type of applications executed on it, different execution strategies might be the solution to energy conservation. For instance, the Race to Halt strategy is proved to be a solution if the CPU is not the major power consumer in the platform [6]. By contrast, if the static power dissipation of the platform is relatively low, the execution with a lower clock frequency of the application might save energy by going slowly to the application results [6].

In this paper, we aim at locating the working configuration in which the overall platform energy consumption is minimized. Computational offloading is the mechanism of moving heavy tasks to more powerful computing units. This mechanism can be a winning strategy for achieving good levels of energy efficiency in the case of highly demanding applications. The goal of this paper is to analyze the impact of computational offloading on modern embedded mobile computing platforms, having multi-core heterogeneous architectures, to achieve platform-level energy efficiency in the case of highly computational-demanding object detection and recognition applications.

The main contributions of this paper are the following:

- We present experimental results including energy ef-

ficiency analysis of embedded environment from the platform energy consumption perspective.
- We demonstrate computational offloading as a solution to help to achieve the goals of platform energy efficiency for embedded computing platforms.
- We analyse the energy cost of different neural networks performing object detection and recognition on a mix of input images.

The rest of the paper is organized as follows. In Section II, we review the related work. In Section III, we develop the discussion of energy efficiency extended to the platform level. In Section IV, we present an offloading strategy aiming at achieving energy efficiency. Section V is explained AI approach for object detection and recognition. Finally, we conclude with experimental testing and numerical result discussion in Sections VI and VII.

II. Related Work

Several works are addressing the topic of energy management and computation offloading in mobile systems. More specifically, offloading compute-intensive tasks towards more powerful systems is a well-known research area, which dates back to the '70s. In [7], the authors present a framework for computation offloading that is based on a comparison of local and remote cost of computations. The decision making part of the framework predicts the communication bandwidth for assessing the costs. In this work, we experimentally validate different communication bandwidths and propose the bandwidth as a metric for the offloading decision. In [8]–[12] authors present offloading techniques that target battery-operated devices with the characteristic of possible offloading of single methods inside the task that needs to be executed. By contrast, we focus on offloading the full task without previously running it on the local or remote side. On the energy management side of the multi-core architectures, [5], [13] show the possibility to maintain low-power dissipation and at the same time to accommodate computation-intensive applications. In [5], authors propose an algorithm that selects the best execution option in terms of energy conservation, depending on the type of application and during runtime the mapping decision might change depending on the phases of the program. On the other hand, we adopt a joint approach of finding the optimal execution configuration inside the heterogeneous architecture and explore the usage of offloading as a mechanism for achieving energy efficiency.

III. Energy efficiency at the platform level

Multi-core heterogeneous CPUs have been introduced in an attempt to increase the energy efficiency of mobile devices, by using the appropriate computing element for the considered task. Complex out-of-order cores are used to handle compute-intensive tasks where performance is central to the outcome, meanwhile, simple lightweight in-order cores provide support for background repetitive tasks that run continuously, yet do not have time constraints. Modern platforms offer a range of different core capabilities: simple cores have simple data-path and low power profile, medium-complexity cores introduce more complex pipeline, and high-complexity cores work with high levels of parallelism and high voltages. The use of high voltages in complex cores comes with highly dynamic power dissipation which will increase the temperature of the chip and also the relative static power dissipation which comes as main side effect [14]. The adoption of actuators in today's MPSoC can certainly provide energy efficiency at the chip level as demonstrated in [15], [16] but cannot have a major effect in the overall platform energy consumption. Other parts of the system like memory and I/O result in high energy expenditure. Lowering the core voltage (V) can make transistors switch slowly which, in turn, forces the frequency (f) of the core to a lower level. This can have a great impact on reducing the dynamic power which is proportional to $f * V^2$. However, it comes with the cost of increasing the running time of a task which may increase the energy consumption of other parts of the platform [17] and, therefore decrease the overall energy efficiency of the platform. In Figure 1 we run a neural network application on a Single-board Computer (SBC) and we measured the energy consumption of the CPU, platform and overall board for different CPU frequencies. As it can be noticed from the results, the best strategy from an energy point of view would be neither to use the low nor the high clock frequencies, but setting the clock frequency in between. The dots represent experimental data while the lines show the polynomial curve fitting.

In our previous work [15] we explored the energy efficiency of the MPSoC in multi-core heterogeneous architectures by using the concept of configuration points, that is a tuple of parameters, such as the type of cores used in computations, number of cores, performance level of cores (DVFS), and utilization level of the task using a core. By selecting for execution a specific class of these points, we can achieve better energy efficiency on the level of the CPU.

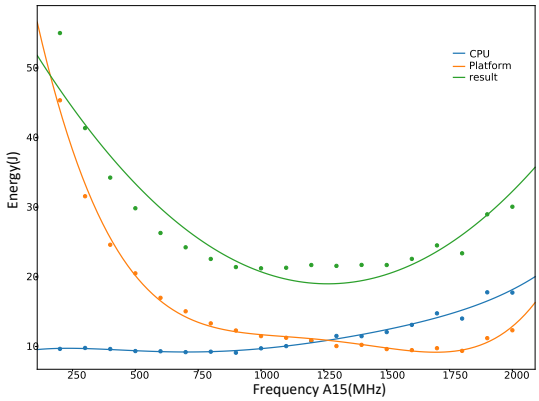

Fig. 1: Measured CPU, base, and total energy consumption of the platform.

IV. Offloading strategy

Despite continuous technological and performance improvements, embedded mobile computing devices come also with

physical constraints, such as processing power and, more importantly, battery life. Many applications including natural language processing, image recognition or online gaming, are challenging in terms of computational requirements. Computational offloading is a mechanism where a task can be sent for computation to a more capable remote unit. As soon as the remote machine has completed the task execution, the results are sent back to the device which initiated the offloading. In turn, computational offloading is expected to be an effective solution to address the data explosion produced by the massive increase of digital services most often run by resource-constraint devices.

A. Application type

The range of applications that are deployed on embedded mobile computing devices, for example in autonomous robots or drones, potentially benefiting from offloading, is very broad. 3-D mapping, speech recognition, object detection and recognition are among the usual candidates for employing computational offloading. Applications change between each other with a diversity of factors e.g., concurrency, which might be at the task level or at the data level as in streaming applications. Emerging use cases are those requiring intelligent image processing in drones for example, where first processing of the captured images could be done on the drones, even though computationally expensive pattern recognition to detect certain special conditions, as fire or floods, could be done in the edge. Another interesting application from Industry 4.0 seeks for achieving zero-defect manufacturing (ZDM), where cameras provide 4k/HD video stream of the goods moving in conveyor belts, and AI services deployed in the edge/fog can provide the intelligence for identifying damaged parts and controlling the robotic arm to sort the material in accordance. In this work, we consider an object-detection application run through a deep neural network. We run different neural networks for object detection on images of different sizes and background complexity. We then define what are the conditions to consider to achieve an optimal solution when offloading tasks.

B. Offloading criteria

Offloading is a natural solution when the application presents real-time constraints that could not be met by the mobile platform. Applications are divided into two parts: one part is executed on the mobile platform and the second part, which requires the majority of computations, is handled by the remote server. The mobile platform could be considered as an interface that handles requests or takes images from a camera, and the remote side might be responsible for the heavy computation. If we denote mobile platform speed as s_l and the application workload as w, the time needed to compute locally would be $\frac{w}{s_l}$. Next, to offload to a remote server, we need to send some input data denoted as d_i over network communication, which has a bandwidth B, to a server, whose speed is denoted as s_r, and receive the results as d_o.

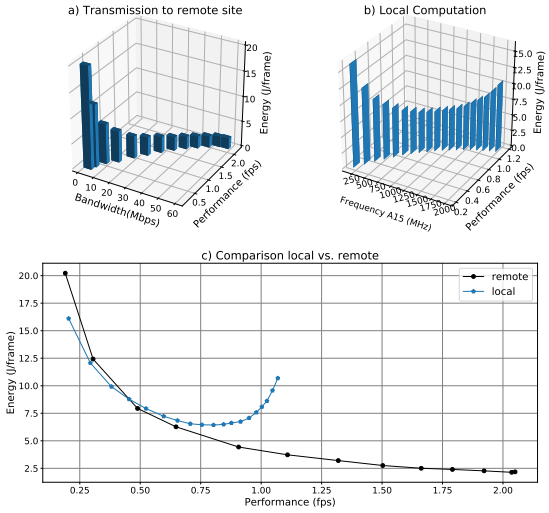

Fig. 2: Energy efficiency of local and remote computation.

The synchronization with the remote side is achieved over TCP protocol and the synchronization time is denoted as t_{tcp}. Overall the time spent to execute the remote computation is:

$$2 * t_{tcp} + \frac{d_i}{B} + \frac{w}{s_r} + \frac{d_o}{B}$$

Offloading increases the performance if the inequality holds:

$$\begin{aligned} &\frac{w}{s_l} > 2 * t_{tcp} + \frac{d_i}{B} + \frac{w}{s_r} + \frac{d_o}{B} \Longrightarrow \\ &w * \left(\frac{s_r - s_l}{s_r * s_l} \right) > \frac{d_i + d_o}{B} + 2 * t_{tcp} \end{aligned} \tag{1}$$

and this can happen in the following cases:

- large w: the amount of computation is considerable;
- $s_r \gg s_l$: the remote side is way faster than local side;
- $d_i + d_o$, the amount of transmitted and received data, is small;
- B is large.

We can also notice that if $\frac{w}{s_l} < \frac{d_i + d_o}{B}$ then offloading will not increase performance even if the remote side is infinitely fast ($s_l \longrightarrow \infty$). In Figure 2 we show the result of performing object detection, using an Single Shot Detector (SSD) neural network as described below, locally by a mobile platform or offloaded to a remote server for computation. In graph c), we present a comparison of the performance vs. energy efficiency reached by the local platform and remote side. We can notice that the mobile platform can achieve performance up to 1.2fps, and if we need an additional frame rate, the computation should be offloaded to the remote side. Again from Figure 2, we notice in graph a) that offloading is energy efficient if the bandwidth of network transmission is as high as possible, on the other hand from graph b) we see that local computation is energy efficient if we use middle core frequencies instead of high or low frequencies.

Another factor that concerns mostly battery-operated devices is energy. Despite technological improvements, battery technology is still far behind the improvements made in other areas of computing systems such as memory, CPU, and screen. Therefore, offloading to extend battery life and conserve energy is of paramount importance, and this criterion can be named as offloading for energy efficiency. If the power to compute task w by the mobile platform is p_l, then the energy consumed by executing the task locally is $p_l * \frac{w}{s_l}$. Otherwise with p_i the power of the system while idling and p_t the power of platform while transmitting, the energy to offload the task can be defined as:

$$p_t * \left(\frac{d_i + d_o}{B} + 2 * t_{tcp} \right) + p_i * \frac{w}{s_r}$$

In this case, the offloading mechanism saves energy if:

$$p_l * \frac{w}{s_l} > p_t * \left(\frac{d_i + d_o}{B} + 2 * t_{tcp} \right) + p_i * \frac{w}{s_r} \quad (2)$$

The same observation applies as before, so: to save energy we should look at the amount of data to be transferred, the type of workload and the available bandwidth of the transmission network.

V. Object detection in images

All object recognition pipelines are usually composed of 3 stages: the detection phase, where objects of interest are detected and located in bounding boxes, feature extraction where the object inside the bounding box is analyzed to extract features that represent it, and the recognition phase, where labels are assigned to each object in the bounding box. In the deep learning approach to object recognition, two components are distinguished: the meta-architecture (or object detection framework) and the base network.

A. Type of neural network meta-architectures

In this paper, we analyzed two different types of commonly used neural network meta-architectures: SSD and You Look Only Once (YOLO).

1) SSD: This general term is used to refer to architectures that use a single feed-forward convolutional network to directly predict classes and anchor offsets. This algorithm eliminates proposal generation and subsequent pixel or feature resampling and encapsulates computations into a single network. This makes the system easy to train and integrates into components that require detection [18].

2) YOLO: This introduces a new approach for the detection phase. Instead of using an image classifier at the end, a single neural network predicts bounding boxes and class probabilities directly from the image. This gives the possibility to optimize end-to-end performance. However, because of the detection approach used in YOLO, small objects or groups of small objects located next to each other might become difficult to detect [19].

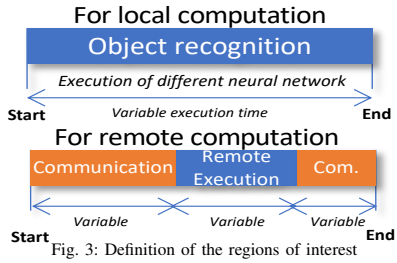

Fig. 3: Definition of the regions of interest

B. Base networks for feature extraction

Several networks, such as Resnet, VGG-16, Inception, Xception, MobileNet, have been developed as base networks performing feature extraction. Some of them, e.g., VGG-16, VGG-19, Resnet-101, have a relatively large memory footprint which makes them impossible to execute on embedded mobile computing platforms, having limited memory capabilities. Other solutions, like Resnet-50 and especially MobileNet, are optimized for running on embedded computing platforms and have therefore smaller memory footprints and computational requirements. In this paper, we evaluate the following four base networks: Resnet, MobileNet, Inception, and Xception, as they are the ones designed to be executed on embedded computing platforms.

VI. Experimental platform and testing

As a central mobile platform, we use an SBC which will run our main application. We run our application on an Odroid XU4 development platform provided by HARDKERNEL. The board is equipped with an Exynos 5 MPSoC, which is an octa-core composed of 4 ARM Cortex A7 and 4 ARM Cortex A15 organized in a big.LITTLE architecture with Global Task Scheduling (GTS). The A7 cores (little cores) can scale up to a frequency of 1.4 GHz, while A15 cores (big cores) can reach a frequency of 2 GHz. The development board runs a Linux OS with kernel 4.2. The board network connectivity is provided by a USB Wi-Fi 802.11n dongle W522U, which is a dual-band wireless USB adapter that provides maximum wireless speed up to 300Mbps over two bands, with a maximum transmit power of 18dBm. The drivers offer two work modalities, one for full power transmission and the other for low power operation while listening for connections. We use both modalities in our experiments. We define our Region Of Interest (ROI) as illustrated in Figure 3. When the object recognition task is executed locally, the ROI is defined as the execution time of the used neural network to recognize an object in the provided input image. When the object recognition task is executed remotely, the ROI is defined from the moment the TCP connection is negotiated with the remote server until the moment the embedded mobile computing platform receives all coordinates, size, and tags associated to all recognized objects.

For measuring the power dissipation inside the ROI, we need an approach that records at a high sampling rate the current consumed during variable execution and communication periods. For example, for the input pictures used in this

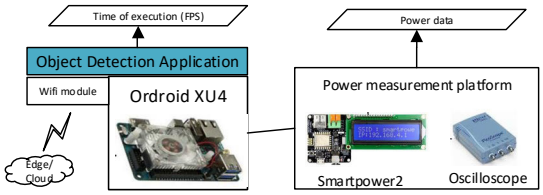

Fig. 4: Schematic of the components of the experiment

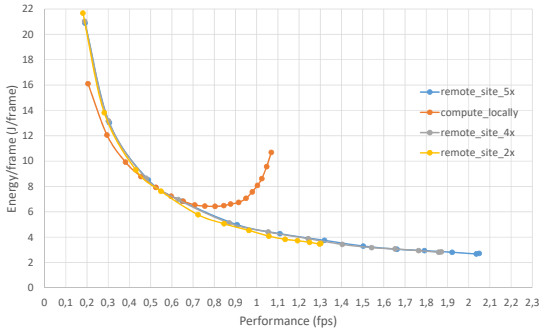

Fig. 5: Comparison of energy spent per frame of local computation versus remote computation when using SSD-MobileNet

paper, the smallest remote execution time is around 200 ms while the longest execution time is around 1100 ms. For the same set of used input pictures, the execution time of the object recognition task on the embedded mobile computing platforms range from 900 ms to 4 s. For setting the boundaries of our ROI, we use one GPIO pin of the board as a flag to measure the start and end the time of the ROI. For measuring the power dissipation, while choosing to offload the task, we use an oscilloscope with one probe recording the current drawn from the board and the other sensing the voltage on the GPIO pin, which is used as a flag for defining our ROI. The testbed is composed of several components as shown in Figure 4.

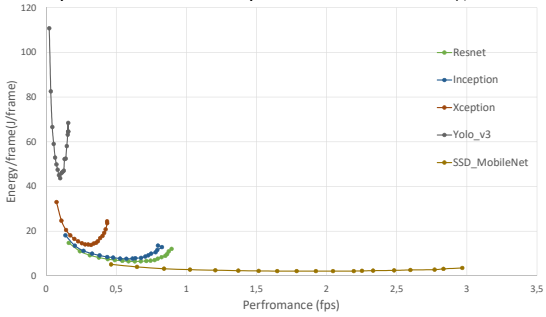

Fig. 6: Comparing the Energy Efficiency of different object detection algorithms.

As a remote side, we use a server that is on the same LAN as the Odroid board and listens for connections by the client. We set the server for different performance levels, such as 5x (five times faster than local side), 4x, and 2x. To measure the power dissipated by the entire board during the ROI we used an oscilloscope and a power meter, depending on the required sampling rate, with measurements logged at a resolution of up to 10 kHz.

The neural network used in the object detection and recognition application is a CAFFE implementation of an SSD meta-architecture with MobileNet [20] as a base network. We tested the change in the neural network used for our object detection. By using neural networks for object detection recognition in embedded environments, the most stringent limitation is the memory available on the device. With only 2 GB of memory, a few networks might be executed on the board. We analyzed three pre-trained base networks: Resnet-50, Inception and Xception implemented in Keras and trained with ImageNet dataset. We used YOLO v3 neural network which is implemented in the darknet framework and trained with COCO dataset.

In the Odroid board, we have different options for running computations related to a certain application. Having 8 computing cores divided into two cluster types, with each type of core being able to work in different performance levels (DVFS), opens possibilities for selecting the optimal energy-efficient way for running the application. We run the object detection application on a single image with all the possible configurations present in our experimental board and we ranked the energy efficiency of each configuration. By reviewing the results of the energy efficiency achieved for each configuration we discover that only the configurations using all four A15 cores, which provide high performance, are ”relevant” for this use case. The best scores in energy efficiency are only achieved by the configurations using all four A15 cores at the middle range frequency. We consider this configuration point as the one we will use in local computations, providing the highest energy efficiency at the platform level.

We executed the neural networks on a set of pictures composed of mix images with high and low resolution, large and small complexity in terms of the number of objects inside, and with different backgrounds. When offloading the detection and recognition task to the remote side, the input images were transmitted with bandwidths from 1 to 60 Mbps.

VII. Numerical Results

Figure 5 shows the average energy consumed to process, using SSD, an image with different execution speeds provided by the remote and local side. We can notice that when using offloading we reach similar energy efficiency for an achieved performance level, regardless of the performance of the remote side. The crossing with local computation is done from 0.5fps to 0.7fps, and for higher performance, the energy cost to perform the detection and recognition task locally is higher than using computational offloading.

We also measured the energy efficiency for processing an image by different types of neural networks. We used a mix of images for testing different neural networks and the results of the average energy consumed to process a frame are shown in Figure 6. As shown, there is a large difference in the results

from each neural network: while we can notice that SSD-MobileNet is the most efficient one in terms of energy, it also provides the highest performance possible on the SBC. Next, we have three other models that provide nearly similar results of energy efficiency: Resnet, Inception and Xception with the first one performing better in terms of energy and achievable performance. Yolo is the one that is energy demanding and has the highest requirements in terms of computations. In [21] authors experiment with different base networks and report the results of the accuracy that the networks provide in terms of the mean Average Precision (mAP) score. They test different meta-architectures with different base models and the results show that SSD with MobileNet is overall the fastest one but provides the lowest accuracy. According to [22], Xception outperforms Inception and Resnet in terms of accuracy. On the other hand, Resnet is less accurate than Inception. Regarding YOLO, according to [23], this network is 20% to 30% more accurate than SSD-MobileNet, but from Figure 6 we notice that it is 170% to 180% less energy efficient. In comparison with Resnet, Yolo is more accurate as described in [24].

VIII. Conclusions and future work

In this paper, we discussed computation offloading as a possible mechanism to reach the optimal platform energy-efficient state considering the energy consumption of all components in the mobile system. As an application test case we considered object detection and recognition performed on several types of images to define when offloading is beneficial to the platform energy efficiency. We identified the configuration points, where the platform provides the maximum energy conservation approach. We concluded that if the bandwidth of the network connection is large enough, then the offloading strategy turns out to be more energy-efficient than the local computing. We surveyed the energy efficiency of different neural network algorithms in an embedded environment and concluded that not many neural networks for object detection can be handled by average embedded platforms. In some cases, to improve the accuracy between 20% to 30%, the cost in degrading energy efficiency is 170% to 180%.

As future work, we consider to extend investigations with more recent MPSoC like Kirin 960 and extend the discussion of platform efficiency by including the GPU for the neural network computations. We also plan to run experiments with GPUs ranging from the ones present in smartphone chips, like ARM Mali-G71 in Kirin 960, to NVIDIA GPUs present in Jetson-TX2 with 265 CUDA Pascal cores and NVIDIA Xavier with 512 CUDA Volta cores.

Acknowledgment

This work has been partially funded by the H2020 EU/TW joint action 5G-DIVE (Grant no. 859881).

References

[1] ericsson.com, "Ericsson mobility report," 2019. https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf.

[2] W. W. R. Forum, "Wireless of big data of smart 5g." https://www.wwrf.ch/files/wwrf/content/files/publications/outlook/White%20Paper%202-%20Wireless%20Big%20Data%20of%20Smart%205G.pdf.

[3] R. Schmoll, S. Pandi, P. J. Braun, and F. H. P. Fitzek, "Demonstration of vr / ar offloading to mobile edge cloud for low latency 5g gaming application," in *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 1–3, Jan 2018.

[4] Y. Li, A.-C. Orgerie, I. Rodero, B. L. Amersho, M. Parashar, and J.-M. Menaud, "End-to-end energy models for Edge Cloud-based IoT platforms: Application to data stream analysis in IoT," *Future Generation Computer Systems*, vol. 87, pp. 667–678, Oct. 2018.

[5] E. Rotem, U. C. Weiser, A. Mendelson, R. Ginosar, E. Weissmann, and Y. Aizik, "H-EARtH: Heterogeneous Multicore Platform Energy Management," *Computer*, vol. 49, pp. 47–55, Oct. 2016.

[6] H. Hoffmann, "Racing and pacing to idle: An evaluation of heuristics for energy-aware resource allocation," in *Proceedings of the Workshop on Power-Aware Computing and Systems*, HotPower '13, (New York, NY, USA), Association for Computing Machinery, 2013.

[7] R. Wolski, S. Gurun, C. Krintz, and D. Nurmi, "Using bandwidth data to make computation offloading decisions," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–8, April 2008.

[8] Changjiu Xian, Yung-Hsiang Lu, and Zhiyuan Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *2007 International Conference on Parallel and Distributed Systems*, pp. 1–8, Dec 2007.

[9] P. A. Rego, P. B. Costa, E. F. Coutinho, L. S. Rocha, F. A. Trinta, and J. N. d. Souza, "Performing Computation Offloading on Multiple Platforms," *Comput. Commun.*, vol. 105, pp. 1–13, June 2017.

[10] K. Kumar, Y. Nimmagadda, and Y.-H. Lu, "Energy Conservation for Image Retrieval on Mobile Systems," *ACM Trans. Embed. Comput. Syst.*, vol. 11, pp. 66:1–66:22, Sept. 2012.

[11] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-Efficient Offloading for Mobile Edge Computing in 5g Heterogeneous Networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[12] A. Bhattacharya and P. De, "A survey of adaptation techniques in computation offloading," *Journal of Network and Computer Applications*, vol. 78, pp. 97–115, Jan. 2017.

[13] M. Malik and H. Homayoun, "Big data on low power cores: Are low power embedded processors a good fit for the big data workloads?," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, (New York City, NY, USA), pp. 379–382, IEEE, Oct. 2015.

[14] S. Variable, "A multi-core cpu architecture for low power and high performance," *Whitepaper-http://www. nvidia. com*, 2011.

[15] H. Rexha, S. Holmbacka, and S. Lafond, "Core level utilization for achieving energy efficiency in heterogeneous systems," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 401–407, March 2017.

[16] U. Gupta, C. A. Patil, G. Bhat, P. Mishra, and U. Ogras, "DyPO: Dynamic Pareto-optimal configuration selection for heterogeneous MpSoCs," *ACM Transactions on Embedded Computing Systems*, vol. 16, p. 123, Sept. 2017.

[17] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating Server Idle Power," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, (New York, NY, USA), pp. 205–216, ACM, 2009.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016.

[19] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[21] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," 2016.

[22] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2016.

[23] D. Holz, K. Genter, M. Saad, and O. von Stryk, *RoboCup 2018: Robot World Cup XXII*. Lecture Notes in Computer Science, Springer International Publishing, 2019.

[24] J. Redmon, "Darknet-53 accuracy comparisons." https://pjreddie.com/darknet/imagenet/#darknet53_448. Accessed: 2020-01-17.

Paper 6

Data Collection and Utilization Framework for Edge AI Applications

Hergys Rexha, Sébastien Lafond

Originally published *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*

Data Collection and Utilization Framework for Edge AI Applications

Hergys Rexha, Sébastien Lafond
Åbo Akademi University
hergys.rexha@abo.fi, sebastien.lafond@abo.fi

***Abstract*—As data being produced by IoT applications continues to explode, there is a growing need to bring computing power closer to the source of the data to meet the response-time, power dissipation and cost goals of performance-critical applications in various domains like Industrial Internet of Things (IIoT), Automated Driving, Medical Imaging or Surveillance among others. This paper proposes a data collection and utilization framework that allows runtime platform and application data to be sent to an edge and cloud system via data collection agents running close to the platform. Agents are connected to a cloud system able to train AI models to improve overall energy efficiency of an AI application executed on a edge platform. In the implementation part we show the benefits of FPGA-based platform for the task of object detection. Furthermore we show that it is feasible to collect relevant data from an FPGA platform, transmit the data to a cloud system for processing and receiving feedback actions to execute an edge AI application energy efficiently. As future work we foresee the possibility to train, deploy and continuously improve a base model able to efficiently adapt the execution of edge applications.**

I. Introduction

Edge computing is a fast-growing technology trend, which involves pushing compute capabilities to the edge. Edge computing can be described as a distributed computing paradigm that brings computation and data storage closer to the location needed to improve response times, save bandwidth, and improve security.

Edge systems are the deterministic embedded communication and real-time control engines that reside at the edge of the network and closest to the physical world of factories and other industrial environments, e.g., motion controllers, protection relays, programmable logic controllers, and similar systems. Clock frequencies in gigahertz, larger memory sizes, higher numbers of input/output ports, and the latest encryption engines might seem to offer solutions for future requirements. However, when dealing with the timescale of industrial equipment, which has critical subsystems that operate on a scale of hundreds of microseconds (or less) and need to operate in factories and remote locations for decades, relying solely on a cutting-edge multicore embedded processor is risky. A much higher degree of freedom in scaling is desperately needed, at for example Industrial edge system, due to the timescales involved. Also there is a need for a more consistent approach that allows determinism, latency, and performance to be easily managed. At the heart of the current industrial revolution is the roll-out of machine learning (ML) algorithms, specifically deep neural networks (DNNs). They achieve impressive results

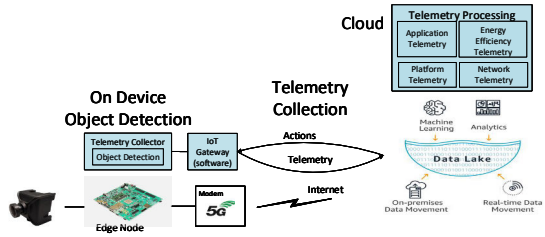

Fig. 1. Basic schematic of the telemetry framework

in computer vision and speech recognition, and are increasingly being adopted for other tasks. DNNs are first trained on a labeled dataset, and afterwards can be used for inference on previously unseen data as part of an application. The large compute and storage requirements associated with DNN deployment necessitate acceleration. Furthermore, different constraints might be imposed on accuracy, cost, power, model size, throughput, and latency depending on the use case. Real-time and safety-critical applications such as augmented reality, drone control, and autonomous driving are not suitable for offloading to the cloud due to low-latency requirements and data transmission overhead. In cloud-computing and ML-as-a-Service contexts, data centers face ever-increasing throughput requirements to process astronomical scales of data [13], bringing additional challenges in energy efficiency to minimize operating expenses. While cloud service latency is less critical compared to embedded scenarios, it still translates directly into customer experience for interactive applications. Traditionally, machine learning research was focused on improving the accuracy of the models without particular regard to the cost of inference. This is evident in the older networks like AlexNet and VGG, which are now considered large and with many parameters [2]. However, as machine learning and DNNs move into practical applications, compute and memory requirements become a major concern.

II. Use case and Architecture Overview

The assumed scenario for this work is the following: an industrial system (it could be for example a patrolling robot or a manufacturing conveyor) is streaming live a video over a 5G network and requests as a service detected objects from the video stream. The object detection service is executed from the Multi-access Edge Computing (MEC) of the used 5G base station.

The main assumptions of the work are the following: (a) an FPGA platform can provide lower latency than more traditionally used GPU platforms for this type of application. This is due to the datapath architecture of the FPGA and DPU, which does not require to first "flood" a large number of Streaming Multiprocessors (SM) as in a GPU; (b) taking advantage of the DPUs, we can reach higher throughputs in terms of number of processed frames per second; (c) the FPGA platform will provide a better energy efficiency solution compared to CPU and GPU based solutions.

The main goals of the work are the following:

- Propose and implement a telemetry collection framework that complements the use case scenario described above.
- Evaluate achievable latency, throughput and energy efficiency of common edge platform alternatives for the selected use case and proposed architecture.
- Show the benefits off customizing computations at the edge with the intelligence from the cloud side created with the collected telemetry data.

A. Telemetry Framework

There is a need for big data analytics and machine-learning based AI technologies for the operational automation of factories and other industrial environments. These use cases deploy edge systems for real-time control of the operations. The collection of large amounts of data is required from different system components like applications, edge platform and network. The single-sourced and static data acquisition cannot meet this data requirements. It is therefore desirable to have a framework that integrates multiple telemetry approaches from different components. The telemetry framework brings a solution to this problem. The main focus of this work is to provide an end-to-end description and evaluation of the proposed telemetry architecture which is described in Figure 1. The framework can be divided in two parts: the edge part which is described on the left of the Figure 1, and the cloud part which is on the right side. At the edge side of the schematic we have a highly heterogeneous platform which is equipped, either with GPU or with re-configurable hardware (FPGA). The platform is hosting an intelligent application which uses a convolutional neural network (CNN) for performing real-time video inference, and an agent which is collecting several metrics from the application, platform, and network called the telemetry agent. Metrics of various components of the platform are collected and formatted as a JSON object and sent to the other part of the framework. On the cloud part the data is analyzed and actions are taken as a feedback controlling the behaviour of the intelligence performed on the edge side. A more detailed description of the telemetry framework is available from [6].

III. Edge platform technologies

To support our assumptions on the achievable latency and performance of FPGA platforms for CNN-based edge applications, we evaluate two different platforms for the role of the edge node: a Nvidia Jetson AGX Xavier (as a representative GPU-based platform) and a Xilinx ZCU102 (as a representative FPGA-based platform). The Xavier is an embedded GPU platform which promise to offer high compute density and good energy efficiency for AI related applications. The Xavier is equipped with 512 CUDA cores with Volta architecture GPU running at 1.37GHz and a 16GB LPDDR4X @ 2133MHz memory with a bandwidth of 137 GB/s, and a flash storage eMMC 32GB. The Xilinx Zynq UltraScale+ MPSoC ZCU102 board has a 16nm XCZU9EG FPGA, an on-board 4GB 64bit DDR4 RAM with a peak bandwidth of 136Gb/s.

We aim at testing the AI inference capabilities, and the power dissipation of the two platforms while running neural network algorithms. The experiments are conducted using Yolov3 and SSDResnet50Fpn algorithms for object detection which perform inference on a 420p video file. In Table I we report the measurements done for both platforms for metrics such as end-to-end delay (EE latency) to process a single frame and number of frames per second processed for a single dissipated watt (FPS/Watt) while running two popular object detection algorithms such as Yolo and SSD. The neural network is fed with the same video file and the power is measured on the entire platform. FPGA architecture is able to achieve good latency in time-sensitive jobs due to the circuit-level customizations on its massively parallel computing units. From the results shown on the table there is a clear advantage

Platform	Algorithm	EE latency (ms)	FPS/Watt
Xavier AGX	Yolov3	120	0,3
	SSD_Resnet50_fpn	250	0,17
ZCU102	Yolov3	29,4	1,48
	SSD_Resnet50_fpn	200	0,37

TABLE I
INFERENCE CHARACTERISTICS OF THE CONSIDERED EDGE PLATFORMS

of the FPGAs platforms versus GPUs to be used especially in streaming applications, this is noticeable in terms of latency and energy-efficiency. The SSD_Resnet_50_FPN is a heavier model compared to Yolo, requiring 178.4 Gops compared to 65.63 Gops of the other side. Based on this evaluation, the ZCU102 FPGA-based board will be used a the edge device in the rest of the paper.

IV. Experimentation Methodology

As described in section II the telemetry framework consist mainly in two parts: the Edge and Cloud side.

A. Edge Side

On the edge side we are executing a CNN-based video inference application which is quantized and pruned for running on a FPGA device. We also collect the parameters which will make up our telemetry data through an agent that is running on the device. The agent collects telemetry data from three categories as described below:

1) Application Telemetry: Latency of the application, FPS.

2) Model Efficiency Telemetry: Computational Unit utilization, Memory Throughput, CPU utilization, Memory utilization, AI model efficiency.

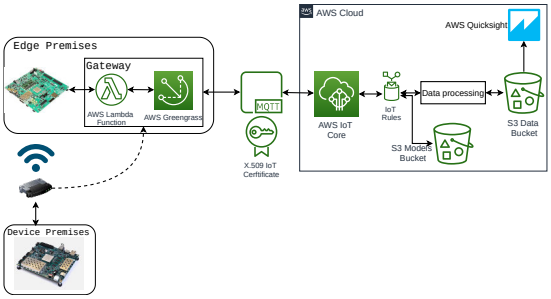

Fig. 2. Proposed Telemetry Infrastructure

3) Energy Efficiency Telemetry: Dissipated power, Temperature of the module, FPS/Watt.

4) Communication Network Telemetry: From the network side the agent collects parameters of the 4G modem used in the experimental framework and these telemetry is sent transparently to the cloud. The parameters collected include: RSSI, RSRQ, RSRP, Temperature, DL/UP.

B. Cloud Side

The second part of the architecture consists of the cloud side which offers services for securely receiving the telemetry data, enriching those data with additional information (e.g. timestamps), analyzing the information contained in the data, providing feedback actions to the edge platform based on some defined triggers and additional services such as further processing, storage, and analytics. The practical implementation used in this paper is based in the Amazon Web Services (AWS) cloud environment.

C. Telemetry Architecture

In Figure 2 we can see the actual components in the deployment of the telemetry architecture for both the edge and cloud sides. In the edge platform we have deployed the AWS IoT Greengrass software which provides the environment for running lambda functions to control the hardware platform and the application running on the platform.

The workflow of the process is as follows: At first on the edge premises the telemetry agent is running and collecting metrics from the application, AI neural network and hardware platform. The collected information is packed into a JSON object and sent to the Greengrass core (GGC) located on the edge side. The GGC is registered with AWS IoT Core on the cloud side and uses the MQTT protocol to forward the JSON objects to the cloud. The IoT Core provides a Device Gateway which manages active device connections and a powerful Message Broker which routes the messages with low latency. Once a message is received we use AWS IoT Rules to send messages to further data processing and aggregation before storing them to the S3 data bucket. Other rules are created to call specific lambda functions on the edge which perform actions like checking the achieved FPS by the object detection application and if the value is above 30 fps, lowering the clock frequency of the platform processing unit in order to save power. Another rule checks for model efficiency, which is the model fps divided by the ratio of peak accelerator rate and model workload, and if the number is below a certain threshold triggers a lambda function on the edge which downloads a new model from a S3 models bucket, located in the Cloud, to the edge premises to perform the inference with the new model. In the telemetry framework in Figure 2 the video stream is transmitted to the edge platform from the device via a 4G or 5G connection. Beside the application, ML model, and edge platform telemetry we also collect network telemetry as explained above, which is sent to the cloud. This data can be exploited for training machine learning models able to predict the connection bandwidth from parameters collected from the router. There are several policies that could decide the location of the inference. By exploiting the telemetry data collected from the router we can predict the bandwidth of the connection and decide whether it is reliable to send the video stream to the edge. There are several research work which show the possibility to predict the current connection bandwidth based on parameters like RSRP, RSRQ, and historic throughput [14, 1, 7]. In the case that the bandwidth is high enough, the stream can be transmitted to the edge premises for faster inference, otherwise the edge will decide to push the inference on the device itself, resulting in slower inference time. The decision on where to actually run the inference in this case will be made on the cloud side based on the received telemetry, and from the model results which predicts the available bandwidth. The offloading decision, from edge to device, could be made by the edge system also, which in case of high levels of utilization can decide to send the intelligent application to the device.

V. Related Work

There is a wide research work regarding the usage of data analytics in making smart and fast decisions especially in wireless networks [4, 8, 9, 10, 3]. Mainly the advances in IoT and hardware/software technology have given the opportunity to collect real-time data from user equipment or core devices which are valuable in making decisions that will impact the performance, adaptability, efficiency of the end-to-end system. This collection of works emphasis more the need for gathering telemetry data from different components of the end-to-end application system. Beside the telemetry data collection there is also to consider the edge component, which in many cases is used to bring resources closer to device side and is a central actor in the real-time applications as in [5, 12, 11]. In this paper we propose a framework that includes different telemetry data, gathered from the edge platform, application, network, and machine learning model with goal of providing feedback to the edge or device premises plus creating a data lake for training machine learning models at the cloud side.

VI. Experimental Results

A. Latency measurements

Devices connect to AWS IoT and other services through AWS IoT Core. Through AWS IoT Core, devices send and receive messages using device endpoints that are specific to

the used AWS account. There are two main communication protocols for sending the data to the message broker in the IoT Core service. One is MQTT, which is a lightweight and widely adopted messaging protocol that is designed for constrained devices, and the other is HTTPS over websockets. To evaluate the proposed architecture, we measured the achievable message latency when reaching the IoT Core and measured any possible difference between the communication protocols. All

Protocol	Mean Lat.(ms)	Min(ms)	Max(ms)	Std. dev(ms)
MQTT	516,44	218	1652	169,45
HTTP	565,75	181	6600	415,91

TABLE II
LATENCY MEASUREMENTS OF SENDING DATA TO THE CLOUD

measurements were done with the Edge platform connected to a commercial 4G network, and the IoT core deployed in eu-west-2 region (London).

As shown in Table II, for the case of MQTT the average latency is lower regardless of the fact that with Greengrass there is an additional delay of the core software. Also the spikes in case of HTTP are quite high bringing a real need for local processing on the edge instead of relying only on the cloud.

VII. Conclusions and future Work

This paper proposes an edge/cloud telemetry collection and utilization framework for applications where reliability, latency, power efficiency and high computational capacity is critical. For instance, vehicle safety as well as vehicular visual and non visual sensing systems could be potential use cases. We evaluate GPU based platform against FPGA platform for the role of edge node in an AI computer vision application and set up our framework with the FPGA platform induced by latency and power efficiency numbers provided. We define the cloud side components of the data lake architecture which will serve later as valuable input for training machine learning networks at the cloud side. At the end we discuss about reaction time of cloud side of the framework and FPGA implementation issues which is good to consider when developing AI-based application on re-configurable platforms. As a future work we foresee the creation of intelligent engines on the cloud side based on the data collected through the telemetry framework.

Acknowledgment

This work has been partly funded by the European Commission through the projects EU-TW 5G-DIVE (Grant Agreement no. 859881).

References

[1] N. Bui and J. Widmer. “Data-Driven Evaluation of Anticipatory Networking in LTE Networks”. In: *IEEE Transactions on Mobile Computing* 17.10 (2018), pp. 2252–2265. DOI: 10.1109/TMC.2018.2809750.

[2] Y. Chen et al. “Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices”. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.2 (2019), pp. 292–308. DOI: 10.1109/JETCAS.2019.2910232.

[3] F. Hussain et al. “Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges”. In: *IEEE Communications Surveys Tutorials* 22.2 (2020), pp. 1251–1275. DOI: 10.1109/COMST.2020.2964534.

[4] P. Jiang et al. “An Intelligent Information Forwarder for Healthcare Big Data Systems With Distributed Wearable Sensors”. In: *IEEE Systems Journal* 10.3 (2016), pp. 1147–1159. DOI: 10.1109/JSYST.2014.2308324.

[5] Z. Khan and J. J. Lehtomäki. “FPGA-Assisted Real-Time RF Wireless Data Analytics System: Design, Implementation, and Statistical Analyses”. In: *IEEE Access* 8 (2020), pp. 4383–4396. DOI: 10.1109/ACCESS.2019.2962200.

[6] Hergys Rexha and Sebastien Lafond. *Data Collection and Acceleration Infrastructure for FPGA-based Edge AI Applications*. 2021. arXiv: 2103.06518 [cs.LG].

[7] J. Schmid et al. “A Deep Learning Approach for Location Independent Throughput Prediction”. In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. 2019, pp. 1–5. DOI: 10.1109/ICCVE45908.2019.8965216.

[8] J. Wang et al. “Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach”. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 2017, pp. 1–9. DOI: 10.1109/INFOCOM.2017.8057090.

[9] L. Wang and S. Cheng. “Data-Driven Resource Management for Ultra-Dense Small Cells: An Affinity Propagation Clustering Approach”. In: *IEEE Transactions on Network Science and Engineering* 6.3 (2019), pp. 267–279. DOI: 10.1109/TNSE.2018.2842113.

[10] L. Wang and S. Cheng. “Data-Driven Resource Management for Ultra-Dense Small Cells: An Affinity Propagation Clustering Approach”. In: *IEEE Transactions on Network Science and Engineering* 6.3 (2019), pp. 267–279. DOI: 10.1109/TNSE.2018.2842113.

[11] Shang Wang et al. “Live Video Analytics with FPGA-Based Smart Cameras”. In: *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 9–14. ISBN: 9781450369282. DOI: 10.1145/3349614.3356027. URL: https://doi.org/10.1145/3349614.3356027.

[12] W. Xia et al. “A Deep Learning Framework for Optimization of MISO Downlink Beamforming”. In: *IEEE Transactions on Communications* 68.3 (2020), pp. 1866–1880. DOI: 10.1109/TCOMM.2019.2960361.

[13] Xilinx. *Xilinx Powers Alibaba Cloud FaaS with AI Acceleration Solution for E-Commerce Business*. URL: https://www.xilinx.com/publications/powered-by-xilinx/xilinx-alibaba-case-study.pdf.

[14] C. Yue et al. “LinkForecast: Cellular Link Bandwidth Prediction in LTE Networks”. In: *IEEE Transactions on Mobile Computing* 17.7 (2018), pp. 1582–1594. DOI: 10.1109/TMC.2017.2756937.

ISBN 978-952-12-4306-6