Universitatea Politehnica Timișoara

LABORATORUL DE CALCUL MOBIL
RETELE DE SENZORI
SISTEME INCORPORATE

Research and Contributions in Energy-Efficiency and Context-Awareness of Mobile Systems and Applications

Habilitation Thesis

Marius MARCU

Timisoara, 2016

CONTENTS

1 SCIENTIFIC AND ACADEMIC RESULTS

1.1 INTRODUCTION AND FIELDS OF EXPERTISE

This habilitation thesis summarizes the achievements I have obtained since when I received the PhD scientific title of Politehnica University of Timisoara back in 2005, in the field of Computer and Information Technology, with the thesis "Cercetări privind fenomenele termice la plăcile cu circuite integrate folosind imaginile termografice şi senzorii de temperatură".

I am currently an Associate Professor PhD Engineer at Department of Computer and Software Engineering, Faculty of Automation and Computers, Politehnica University of Timisoara (http://www.cs.upt.ro/~mmarcu). For almost 20 years of didactic, scientific and research activities, I have addressed and worked in several fields of interests, gaining both theoretical and practical strong experience, as follows:

- Power and Energy Measurement, Modeling, Optimization and Low-Power Design of Embedded Systems
- Dynamic Thermal, Power and Energy Management of Computing Systems
- Microprocessor Systems Design and Microprocessor Architectures
- Computer Networks and Wireless Communication
- Mobile Computing, Systems and Applications
- Low Level Software and Drivers Development

In these fields of expertise, I have published over 70 scientific and academic works as single author (9), first author (34) or co-author:

- 1 PhD thesis and 2 PhD reports;
- 3 books and 3 books chapters;
- 6 online course and laboratory support materials and 1 printed laboratory workbook;
- 14 scientific papers in journals indexed by recognized databases;
- 53 scientific papers in proceedings of international and national conferences indexed by recognized databases;
- 2 published national patent requests and 1 international patent request.

Since 2005 I have been actively involved in 13 international and national projects granted by competition, as director or principal investigator (2), local partner responsible (2) and team member. I have been also involved in 8 research and development projects with the local industry as director (4) and team member.

- 7 international grants (2 as local partner manager);
- 6 national grants (2 as principal investigator);
- 8 contracts with industry (Alcatel-Lucent).

Since 1995, I worked together with Prof. Mircea POPA to develop the microprocessors laboratory that evolved in time into Research Laboratory on Mobil Computing, Sensor Networks and Embedded Systems (CMRSSI – http://research.cs.upt.ro/node/224). Our research objectives are:

- Research in the areas of mobile computing, smart sensors, sensor networks and embedded systems; development of scientific and industrial applications.
- Developing partnerships with academia and industry at a local, national and European level.
- Continue to increase the laboratory's visibility through scientific publications, patents and creation of new products.

The strategy we use to achieve the proposed objectives are:

- Attracting research funding based on national and international competitions.
- Focus the scientific research on domains and subdomains with a maximum potential by taking into account the actual areas of interest of the laboratory members.
- Orientation on applied research and development tailored to industry needs in the area, increasing collaboration with research groups and development companies.
- Integrating laboratory research in the European Research Area.
- Identifying and establishing new relationships with specialized research groups within universities from the European Union and beyond.
- Provide teaching support, courses and specific teaching material.
- Creation of specialists in these areas by contributing to teaching and student involvement at all levels, bachelor, master and doctoral programs in research

Since 1995 I introduced and developed several courses and laboratories:

- Microprocessor systems (developed course and laboratory)
- Device drivers development (new course and laboratory)
- Mobile systems and applications (new course and laboratory)
- Computer networks administration (new course and laboratory)

Since 2006 I supervised and trained UPT teams for Imagine Cup International student competition. In 2009 and 2010 UPT teams won the national Embedded Development section, being selected for final worldwide competition where we reached the Top 15, respectively Top 6 teams worldwide, out of 300 other teams. In 2012 we won the national Software Design section, reaching the Top 20 teams worldwide.

1.2 OVERVIEW ON THE MAIN SCIENTIFIC ACHIEVEMENTS

1.2.1 Power-aware mobile applications

Power-aware and energy-aware mobile applications analysis, modeling and development research work is part of the Dynamic Power and Energy Management of Computing Systems field of expertise. I started this research work right after I finalized the PhD with the innovative idea at that time of involving high level applications in dynamic power management techniques. This research direction has been supported by two national grants I have managed between 2006 and 2011. The most recent one has been a National R&D Funding Grant (PNCDII-IDEAS), called "Open Execution Framework for Low Consumption Battery Powered Mobile Devices" and it had been developed during three years: 2009-2011. The projects have been initially granted with more than 200k EURO, but due to the crisis in 2008 the budged cuts reached almost 50% of the budget initially approved. The main outcome of this research is an Execution Framework for Power-Aware Applications on Battery Powered Mobile Devices. This research topic converged in recent years into cross-layer power and energy management techniques.

Power-aware applications are software applications that implement application specific power management algorithms in order to reduce and optimize the energy consumption of the system while running them. The main goal of this research effort was to promote power consumption management and optimization of mobile and embedded systems at higher abstraction layers of such systems. The main outcome of these projects was to establish a general theoretical background and applicative rules and patterns in order to obtain efficient mobile systems and applications from the point of view of the consumption. During the last years there had been a lot of research effort in power and thermal management at different abstraction levels of the system such as hardware, drivers and operating system (OS). However, my idea has been to focus on a different direction: the higher abstraction levels of the system - the user applications, due to limited energy efficiency gain achieved by the lower layers of a computing system. Power management, when implemented at the highest levels, leads to a much more efficient power saving than when used in the lower system layers. Still, the large majority of literature deal at that moment with the physical (hardware) and operating system levels. Therefore, the research goal has been modeling, monitoring and controlling of software applications power consumption for different types of hardware platforms.

The results achieved so far in this research direction are both theoretical and applicative:

1. Definition, implementation and validation of two new types of benchmarks: power benchmark and thermal benchmark. First of all, there was a lack of a runtime method which can be used to evaluate energy-efficiency of software applications or various algorithms implementations executed on different hardware systems. Therefore, I introduced and defined the new concept of power-thermal benchmarks software as a software application used for run-time system-level energy characterization. This concept has been first introduced in 2006 and has been further refined in 2007-2009.
2. Proposal of a high-level model for software applications power consumption based on monitoring, profiling and estimation. For the second step, the modeling, instrumentation and

energy consumption estimation for the user applications and their impact on the overall system energy consumption have been addressed. A method to estimate energy consumption of software application has been proposed. The power consumption modeling problem of computing systems is in general a very complex one because each physical component from the system has its own energy consumption which varies with the workload executed by the component. Therefore together with the physical components, the software applications have a big influence on the system's energy consumption. To overcome these limitations, I proposed the decomposition of the whole system in components. The energy consumption of the application has been computed based on estimated for these components. Thus, I proposed a general model based on the system decomposition into hardware and software components that can be monitored, instrumented and evaluated separately.

3. Power characterization experiments on several hardware components or algorithms: CPU multi-core and multi-threading, WLAN communication, multimedia algorithms, and I/O disk operations. The next step has been to run multiple experiments for energy profiling of various hardware components in a mobile device and of different algorithms implementations. These experiments provide a large database of energy consumption profiles of various interfaces, components, algorithms and devices.
4. Development of an execution framework for power-aware software applications. In order to implement power-aware application, I proposed and successfully implemented a power efficiency framework which addresses some of the shortcomings of existing approaches towards power aware software. The framework architecture and its theoretical model in the next section.

1.2.2 Energy profiling of virtualization solution

Virtualization is one of the main research directions for both large scale data centers and servers. Eenergy efficiency has become an important aspect in data centers and large server systems. Our research effort explores how virtualization influences the power consumption of both physical systems and virtual systems and which is the most efficient way to implement such applications.

The main goal of this work has been the study on the power consumption impact of virtualization solutions for common desktop and laptop computers. This work explored how virtualization influences the power consumption of both physical systems and virtual systems and which is the most efficient way to implement such applications.

The main contribution to the project is the study on energy and thermal efficiency of virtualization solutions implemented on the two OS used today: Windows and Linux. In order to achieve this result the evaluation methodology and measurement setup have been proposed and implemented.

1.2.3 Device and execution characterization using power signatures

Using smart grid for developing intelligent applications is a current trend of great importance. One advantage lies in the possibility of direct monitoring of all devices connected to the electrical network in order to prevent possible malfunctions and manage (optimize) the overall power consumption in office or residential buildings. In our work we aim to monitor domestic devices in order to automatically extract their workload power profiles in order to identify bad usage habits,

external conditions or aging and finally failures of these devices. The main goal is to detect and model the influence of the identified elements on the energy efficiency of the devices. In our approach we monitor the target devices in order to extract power consumption profiles of different applications, workload, external conditions or usage patterns. Next we apply pattern recognition algorithms to extract relevant behavior or patterns in the power consumption series of measurements. We call the identified patterns in the power profiles, power signatures. Based on these signatures we further try to count device specific usage parameters and to identify abnormal behavior of the device due to aging or malfunctioning. The final goal of our work is to propose an intelligent house level centralized solution to monitor, analyze, predict and control household devices in order to improve their life time usage energy efficiency.

There are several contributions claimed by our work:

- An intelligent power metering infrastructure deployed as a wireless sensors network for energy monitoring, analyzing, controlling and predicting of electric, electronic and computing devices;
- Definition, implementation and validation of the concept of power signature applied to run-time operation of home appliances and computer devices;
- A database storing power signatures of many kinds of devices, such as refrigerator, freezer, washing machine, bread cooking machine, TV set, DVD player, and desktop computers;
- A method using digital processing and pattern matching techniques for characterizing the functional parameters of the components, user operation modes and patterns, system aging or failures.

1.2.4 Wireless positioning of mobile devices

Location-aware mobile applications is part of Mobile Computing, Systems and Applications field of expertise. The main research objective of this research topic represents the possibility to use the wireless network infrastructure (i.e. networks adapter, access points, and wireless routers) to determinate indoor position of a mobile system, while using the same infrastructure for the wireless communication. This research work has been performed between 2007 and 2009, but a revival of the topic happened after 2010, along with the spread of smart devices and context-aware applications. We did not received any founding to perform this activity. This research topic converged to context-aware services development for mobile users.

Indoor positioning is an important aspect of mobile applications in order to develop context-aware applications. The main goal for localization process is estimating the position of a mobile device in its indoor and outdoor environment based on a set of sensors with known positions. Modern smart mobile devices have one or more wireless communication interfaces, like Bluetooth or WiFi, in order to communicate with infrastructure computing systems or other mobile devices. WLAN infrastructure is a widely accepted and implemented communication standard in many indoor environment, therefore many buildings are already equipped with latest IEEE 802.11 WLAN infrastructure access points. Wireless adapters the modern mobile devices are equipped with can monitor radio power strength of the emitting sources nearby. Based on the received radio signal strength an estimation of distance between the device and power source can be computed. If a

mobile device can monitor at least three WLAN power signal sources with a-priori known positions, an estimation of the mobile device position can be achieved. This approach has the advantage that no costly hardware installations are necessary inside the building, nor additional sensors to be attached with the mobile devices.

The results achieved until now are both theoretical and applicative:

- Proposal of a theoretical mathematic model for position estimation of a wireless receiver based on the received radio signal strength (RSSI) from several access points with known positions and using trilateration.
- Proposal and validation of an extended range of test cases used to evaluate local positioning system deployments. A complete set of test cases that can be used to evaluate a wireless positioning system (WPS) has been proposed. A lot of experiments have been performed in order to build a radio signal strength database. Analyzing the RSSI data, a set of problems that can make such a WPS system ineffective for high accuracy localization has been discussed.
- Development of software module for estimating the positions based on the radio parameters. Based on the proposed model and the test execution results a WPS has been implemented and it can be used to provide location specific services for mobile applications. An accuracy of 2-3 meters has been obtained with current implementation.

1.2.5 Component level power characterization of embedded systems using FPGA designs

Power and energy profiling of multi-core embedded SoC designs is a daunting task due to the lack of fine grain accurate and high sampling rate monitoring infrastructures. Furthermore, shared components used in common by multiple processing cores, make SoC designs difficult to analyze the thread level energy consumption. The main goal of our work is to perform both hardware and software accurate profiling of any multi-core architecture using low level profiles of its components, such as CPU cores, memories, and buses. Therefore, in this section, we investigate the capacity of FPGA designs to profile the power consumption and energy usage of multi-core embedded architectures and application benchmarks running on them.

The main outcomes of this research direction are:

- Implementation and validation of a multi-core target design used to monitor per-component and per-thread energy consumption. The system is heterogeneous and includes two ARM cores and 8 Microblaze cores
- Introduction of a monitoring methodology for fine-grain and high-sampling rate energy measurement, validate by measurements and simulations
- Energy profiling of FPGA core, BRAM memory, DRAM memory, CPU cores, and AXI bus

1.2.6 Runtime Thread Level Energy Accounting

The ever-growing need for energy efficient computation requires adequate support for energy-aware thread scheduling that offers insight into a systems behavior for improved application

energy/performance optimizations. Runtime accurate monitoring of energy consumed by every component of a multi-core embedded system is an important feature to be considered for future designs. Although, important steps have been made in this direction, the problem of distributing energy consumption among threads executed on different cores for shared components remains an ongoing struggle. We aim at designing a generic low-cost and energy efficient hardware infrastructure which supports thread level energy consumption monitoring of hardware components in a multi-core system. The proposed infrastructure provides upper layers (operating system and application threads) with per thread and per component energy accounting API (Application Programming Interface), similar with performance profiling functions. Implementation results indicate that the proposed LEM (Load and Energy Monitor) adds around 10% resource overhead to the monitored system. Regarding the power estimates, the one derived by LEM achieve a correlation degree of more than 95% with the ones obtained from physical power measurements.

The contributions of this work are as follows:

- hardware infrastructure for dynamic energy consumption monitoring in a heterogeneous multi-core system with per-thread energy accounting;
- energy interrupt specification and design;
- a use case on the software side (OS and drivers) for run-time per-thread energy accounting implementation on FPGA; and
- validation of proposed infrastructure on a high-end FPGA board with physical energy measurements.

1.3 OVERVIEW ON ACADEMIC AND EDUCATIONAL RESULTS

Since 1995 I introduced and developed several courses and laboratories:

- Microprocessor systems (developed course and laboratory)
- Device drivers development (new course and laboratory)
- Mobile systems and applications (new course and laboratory)
- Computer networks administration (new course and laboratory)

Since 2006 I supervised and trained UPT teams for Imagine Cup International student competition. In 2009 and 2010 UPT teams won the national Embedded Development section, being selected for final worldwide competition where we reached the Top 15, respectively Top 6 teams worldwide, out of 300 other teams. In 2012 we won the national Software Design section, reaching the Top 20 teams worldwide.

Each year I supervised students for their diploma or master projects. In these collaborations I focus on both technical quality and applicability of the results.

1.4 OVERVIEW ON INNOVATION AND ECONOMIC IMPACT

In my projects I constantly try to focus on the applied results and innovation. I am looking also at the economic impact or the research outcomes. As a result of these interests, in 2012, together with a company in Timisoara (Lasting System) and one of my former students we founded a startup (Intuitive Software SRL) based on the results achieved during the collaboration on diploma and master projects. Our startup was granted with 200 000 EURO from POS-CCE Structural Founds.

The main goal of the product we targeted is an innovative context-aware communication manager. In the last years there is a merging trend towards unified communication systems over IP protocols from desktop to mobile devices. Nowadays users have the possibility to choose from a wide range of methods of communications (voice, text, video, social networks) and equipment (mobiles, SIP phones, computers). This variety leads to a series of problems like the management, synchronization, updating and sharing of contacts between various platforms or choosing the most efficient mean of communication from a cost and quality perspective. In this context the current project has as main goal to propose an intelligent software platform for assisting users of multiple communication technologies (mobile, IP phone, email, social, and messenger) and finding the most efficient method of communication taking into account the available communication channels and the caller and receiver's context. The term of context, nowadays largely discussed in mobile applications developers' community, is any information that can be used to characterize the situation of an entity. The aspects of context that shall be taken into account when proposing a communication channel are: location, people and devices nearby and changes in the previous aspects. Therefore, the proposed application will allow the management of available communication channels along with the contacts in order to optimize the proposal of the most efficient communication method from the quality and cost perspective using both caller and callee contexts. The expected results of the project are: (1) a model for characterizing quality level of communication using context information, (2) an algorithm for selecting the optimum communication channel using machine learning and data mining and (3) a prototype of the proposed platform.

2 EXECUTION FRAMEWORK FOR POWER-AWARE APPLICATIONS ON BATTERY POWERED DEVICES

2.1 OVERALL DESCRIPTION AND RESEARCH OBJECTIVES

Mobile wireless devices continue their fast seizure of our daily lives as they become smarter, powerful and faster. However, the new features built in these devices come with an increase of resources usage and energy consumption. Therefore, prolonging the battery lifetime of mobile devices was one of the main research topics in mobile computing during the last decade. Different power management strategies have been proposed and implemented during the last years in order to reduce the power consumption of the battery powered devices. These strategies can be implemented at different levels of the computing system: physical or hardware level, operating system level, application level and user level. Application-level energy efficiency mechanisms are considered the next important step in the evolution of highly power and energy efficient mobile systems. The final user is also an important element to be introduced within the energy efficiency and management cycle. However the normal user lacks the basic understanding of the low-level power management mechanisms, even though he is the most able to choose how to consume the remaining energy within the battery. Therefore power management techniques should propagate to higher levels of the system, more specifically to the user applications level.

The main goal of this research work was to define, implement and validate a software execution framework for mobile systems (smartphones, tablets and notebooks) in order to reduce and optimize dynamically the energy consumption at the application level. The role of the framework is to provide a generic interface for user applications such that they can obtain an estimation of their own energy consumption and next they should be able to adapt their execution in order to achieve an optimum energy consumption level. Applications that are going to use the execution framework in order to optimize their energy demands are called power-aware applications. The roles of the power-aware application models are first to make the applications aware of their own power consumption and to adapt themselves to reduce consumed power and second to promote and translate system's power management states and actions to the user in a user friendly way. This way the higher-level applications make the users aware of application specific energy consumption aspects and allow users to actively interact with power management mechanisms in a much more effective and informed way.

2.2 THEORETICAL CONCEPTS

In this section we introduce the core concepts, which are used by the framework. We start by introducing theoretical concepts of power sources or power stimuli and responses for different real components as well as measurement and framework specific notions. Next, the different ways we tried to obtain the best solution proposal are discussed together with the advantages and disadvantages of the evaluated solutions.

We define a computing system **S** as a number of hardware components (**C_1-C_n**) together with the running software applications (**A_1-A_m**) using these components (Eq. 1). Every application involves several hardware components in finalizing its tasks in order to provide the services the application is executed for.

$$S = \bigcup_{i=1..m} A_i \oplus \bigcup_{j=1..n} C_j \qquad \textit{(Eq. 1)}$$

The ⊕ operator specifies the usage relation between applications and system components. In fact the relation describes how each component is used by a running application (Fig. 1).

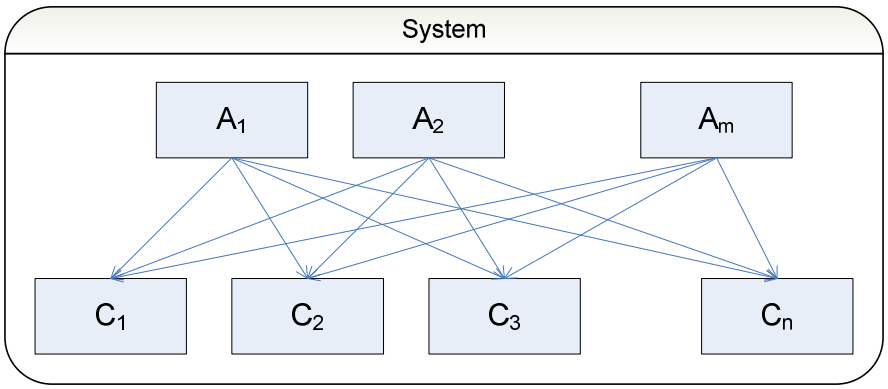

Fig. 1 General system abstraction

In the context of the power efficiency framework, we address different power consumption sources, corresponding to the different components of the system (**C_1-C_n**). Every component **C_j** implements a number of power states (**$S_{j,k}$**), that are activated according to the performance requirements of the applications using it (Eq. 2). Every power state of a component has associated a certain power consumption level (**$P_{Sj,k}$**). This level is considered to be known based on component's datasheets or it can be computed or estimated based on specific formula or by physical measurements (**$M_{j,l}$**).

$$S_j = \{S_{j,1}, S_{j,2}, ..., S_{j,Ns_j}\} \qquad \textbf{(Eq. 2)}$$

Power consumption model of component **C_j** residing in state **S_k** is characterized by the workload performed by C_j. It can be estimated by the set of measurable parameters **$M_{j,l}$** (Eq. 3).

$$P_{S_{j,k}} \approx f_{j,k}(M_{j,1}, M_{j,2}, ..., M_{j,Nm_j}) \qquad \textbf{(Eq. 3)}$$

During its execution every application (**A_i**) uses some components in certain power states or in a certain sequence of power states (Eq. 4). We consider **$S_{j,k}$** the component **C_j** working in power state **S_k**. A component could be placed only in one power state at a time, no matter how many applications are using it. We define power consumption of a software application to be the sum of power consumption levels of components used by the application (Eq. 5). Considering that only one application is running on the system and each component is running in only one power state,

power consumption of one application $\mathbf{A_i}$ is the sum of power levels associated with the used components' power states (Eq. 5).

$$A_i = \bigcup_{j=1..n,k=1..Ns_j} C_j \oplus S_{j,k} = \bigcup_{j=1..n,k\in\{1..Ns_j\}} S_{j,k} \quad \textbf{(Eq. 4)}$$

$$P_{A_i} = \sum_{j=1..n} P_{C_j} = \sum_{j=1..n,k\in\{1..Ns_j\}} P_{S_{j,k}} \quad \textbf{(Eq. 5)}$$

Power consumption of system components and software applications is not constant in time and depends on workload variations that occur during the execution (Eq. 6)

$$A_i(t) = \bigcup_{j=1..n} S_j(t) \quad \textbf{(Eq. 6)}$$

where $\mathbf{S_j}(t)$ means the power state of the component $\mathbf{C_j}$ is at the time t and $\mathbf{S_{j,k}}$ represents discrete power levels. As time passes the component switches between these states:

$$S_j(t) \in \bigcup_{k\in\{1..Ns_j\}} S_{j,k} \quad \textbf{(Eq. 7)}$$

For each power consumption source we can establish different power profiles (or power fingerprints) that represent the variation of power consumption of the component for a given utilization profile (e.g. applied stimuli). Every power profile is identified by a sequence of power consumption values corresponding to component's execution power states associated to the current workload. A component usage model assumes a certain workload usage level and operation for that component. Power consumption of an application $\mathbf{A_i}$ varies in time and could be computed summing the power levels of the components used by the application in every moment in time (Eq. 8)

$$P_{A_i}(t) = \sum_{j=1..n} P_{C_j}(t) = \sum_{j=1..n,S_j(t)\in \bigcup_{k\in\{1..Ns_j\}} S_{j,k}} P_{S_j}(t) \quad \textbf{(Eq. 8)}$$

where $P_{A_i}(t)$ is the power consumption of the system due to the running application A_i at the moment *t*.

Considering that a component could be used by more than one application at a time or a component is not used by an application, we introduce in equation (Eq. 8) a weight factor ($\mathbf{W_{i,j}}$) that specifies the level a component ($\mathbf{C_j}$) is used by an application ($\mathbf{A_i}$) at the moment *t* (Eq. 9):

$$P_{A_i}(t) = \sum_{j=1..n,S_j(t)\in \bigcup_{k\in\{1..Ns_j\}} S_{j,k}} (W_{i,j}(t) \cdot P_{S_j}(t)) \quad \textbf{(Eq. 9)}$$

In terms of energy, the energy needed by an application $\mathbf{A_i}$ to complete its tasks can be computed using (Eq. 10):

$$E_{A_i} = \int_{t=0}^{\infty} P_{A_i}(t) = \int_{t=t_{i,start}}^{t_{i,stop}} P_{A_i}(t) \quad \textbf{(Eq. 10)}$$

where $t_{i,start}$ and $t_{i,stop}$ are the time when the application is started and respectively the time when application is finished.

The power consumption of the whole system at the time *t* can be easily computed by summing the power consumption of every running application in the system including the operating system services and applications (Eq. 11). Summing the power consumption of each component we should obtain also the power consumption of the whole system at the time *t*.

$$P_S(t) = \sum_{i=1..m} P_{A_i}(t) = \sum_{j=1..n, S_j(t) \in \bigcup_{k \in \{1..Ns_j\}} S_{j,k}} P_{S_j}(t) \quad \textbf{(Eq. 11)}$$

Based on (Eq. 11) we aim to assign the overall system power consumption first to the systems' components and second to the running applications. Power consumption distribution to components and applications is further based on the measurements available for components (M_{Cj}) and applications (M_{Ai}).

Considering that for the most battery powered devices, $\mathbf{P_S}$ can be measured at system runtime through well-defined OS API or battery device drivers, we aim to split the overall measured power consumption between the running applications and estimate the power consumption of each application ($\mathbf{P_{Ai}}$). If we can compute and estimate $\mathbf{P_{Ai}}$ from $\mathbf{P_S}$ then we can estimate the effect of every running application on the overall system power consumption. Based on this estimation we can build software applications that are aware of their own power consumption and further these applications can adapt themselves in order to optimize the overall power consumption. A power-aware application is an application that implements application specific power management algorithms.

To dispatch the overall power consumption values we use the power levels of the components' power states associated with user applications (Eq. 8) and the system and components' measurements (Eq. 3) which will be further described in this chapter.

$$P_{C_j}(t) = P_{S_j}(t) \approx f_{C_j}(\bigcup_k M_{j,k}(t), \bigcup_k P_k(t)) \quad \textbf{(Eq. 12)}$$

$$P_{A_i}(t) = \sum_{j=1..n, S_j(t) \in \bigcup_{k \in \{1..Ns_j\}} S_{j,k}} (W_{i,j}(t) \cdot P_{S_j}(t)) \approx \sum_{j=1..n} (f_{A_i,C_j}(\bigcup_k M_{i,j,k}(t), \bigcup_k P_k(t))) \quad \textbf{(Eq. 13)}$$

Based on (Eq. 12) and (Eq. 13) we investigate the possibility to estimate power consumption of system's components and running applications based on a number of measurements we can achieve for every component ($\mathbf{M_{j,k}}$) and application ($\mathbf{M_{i,k}}$). Functions $\mathbf{f_{Cj}}$ and $\mathbf{f_{Ai,Cj}}$ are determined in the process of system profiling where the relation between measurements and power values are established. Due to the complexity of these relations, we focused our research experiments in three directions and try to profile and analyze the CPU and WLAN components and one kind of mobile application - multimedia applications.

2.3 FRAMEWORK DESIGN AND IMPLEMENTATION PROTOTYPE

We propose to implement the power dispatching functions in a software application, we call power framework, which provides a programming interface for user applications in order to permit them to have knowledge of their own power consumption. Therefore we describe next the concepts used to define the power framework module. The power framework (Fig. 2) is placed on top of the operating system services and drivers. It uses the components' device drivers in order to provide a generic and portable interface for the applications running on top of this framework. User applications either register themselves in the framework or the framework monitors them continuously so that it can estimate their power consumption and provide feedback with these power values.

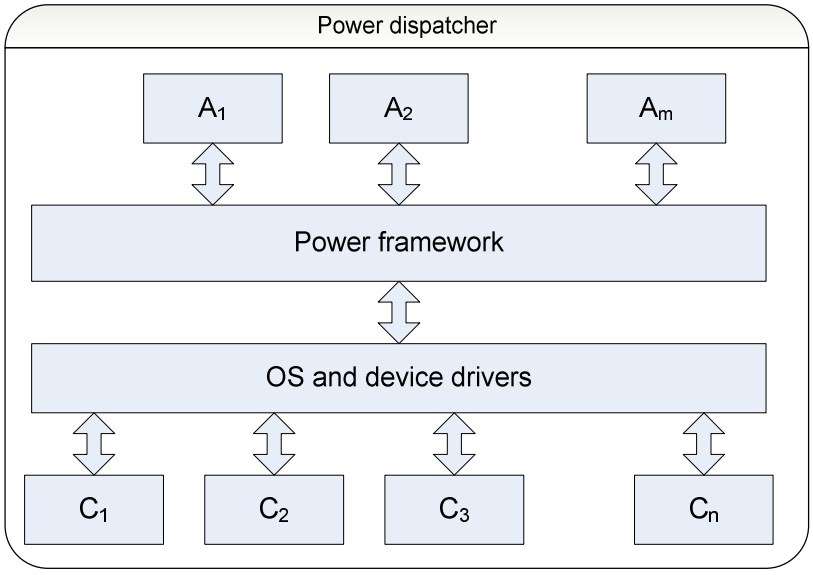

Fig. 2 Block diagram of the power dispatcher

The proposed power framework implements the theoretical concepts presented before and some software specific concepts which are presented further in this section. The general architecture of the framework application is based on a modular structure grouped in several abstracting layers (Fig. 3). At the lower level the framework application will use the operating system's drivers or API functions to access the hardware components considered in the optimizing process of energy consumption: the processor, the battery, wireless chipset, main-board chipset, the memory etc. The framework should be platform agnostic, so that it can be easily ported on different operating systems and hardware devices, therefore the framework core access to operating system and components' drivers is realized through a porting layer.

Generic components' monitors are placed on top of the porting layer so that they provide platform independent measurement functions for the framework core to monitor all available system's components. The components' monitors implement a generic interface in order to permit upper layers to access different types and number of system measurements.

The kernel of the execution framework collects all the available measures through the monitoring drivers and generic components' monitors, calculates the energy consumption of the running

applications and through the application interface, it communicates with registered user applications using specific messages for consumption control.

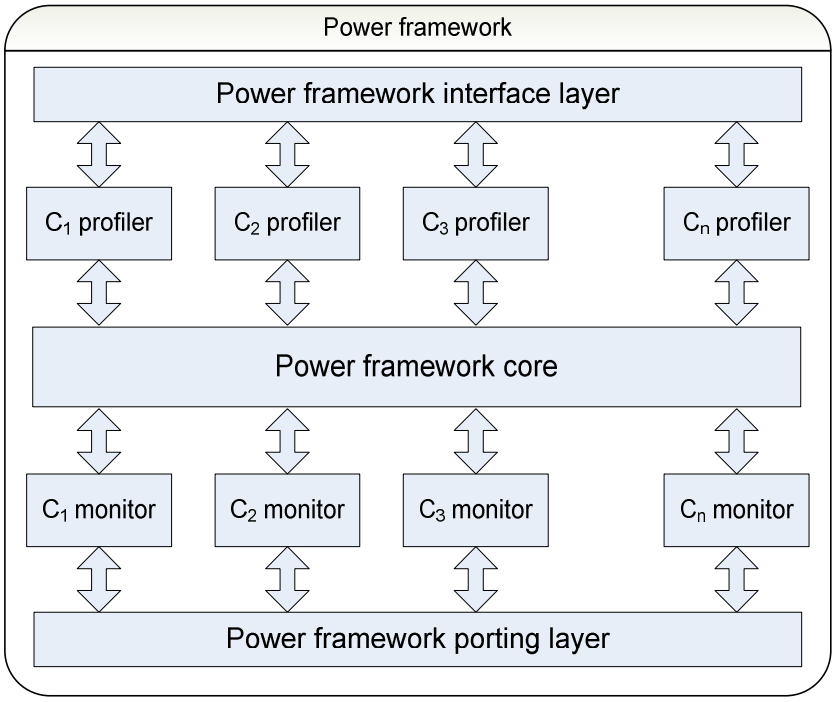

Fig. 3 Power framework abstraction

Further in this section we discuss the system parameters that can be used in developing power-aware applications or power management framework for such applications. We define as power framework metrics the set of parameters that may be used by power management strategies and algorithms implemented to support power-aware applications. There are two layers of power-aware metrics (Fig. 4):

- Component measurements - are the primary values the power-framework will use in order to characterize the current power consumption of the system. These measures can be obtained from different sources: battery driver, operating system, and other device drivers;
- Power efficiency metrics - are the final metrics computed out of the physical measurements, which the power framework will provide to the application level. These metrics are describing high level profiles of different system components that may provide an application level or component level image of the power consumption.

The main goal of the power efficiency monitoring framework (Fig. 4) is to provide applications with power efficiency self-monitoring services. It provides applications with a feedback loop on their power and energy consumption in order to allow applications to adapt their algorithms and tasks to increase their energy efficiency. The data monitoring interfaces of the framework provide energy efficiency measurements based on the framework metrics. Power efficiency measurements can be used to assess the energy efficiency of the platform in terms of different workloads that are applied. The feedback loop is designed in order to improve the power efficiency dynamically, by enabling applications or the operating systems to take dynamic decisions at runtime based on

available energy. For example, in case of a wireless application, the application might adjust the download rate or the wireless driver might adjust the power of the emitted signal, or at least announce the user that the download cannot be finished in the available energy.

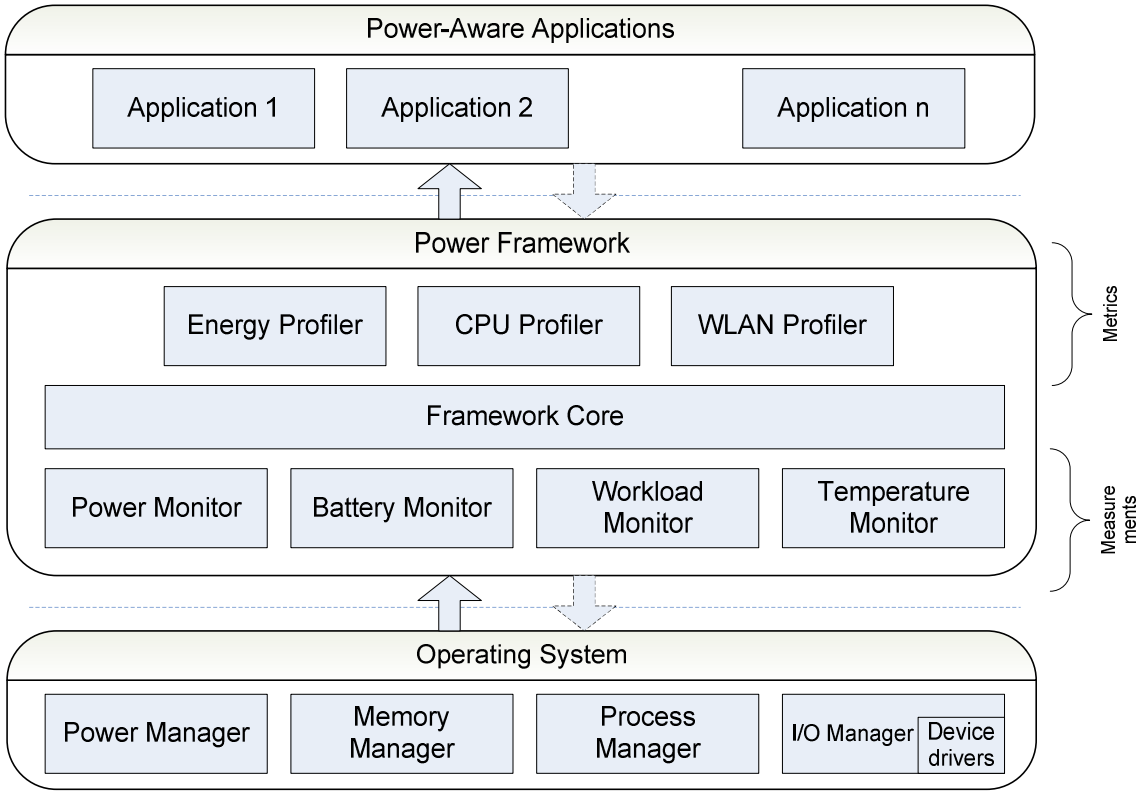

Fig. 4 Power framework modules and metrics

The main goal of the power efficiency monitoring framework at system level is to allow applications and operating system or drivers to improve the both applications' and system's power and energy efficiency. One of the main design requirements of the framework has been its portability. The framework should be ported on different hardware and OS systems easily and potentially at different layers of the system. For example, it can be placed directly in the context of the virtualization layer or can be ported on the guest operating system. Porting the framework at different layers of the overall system is realized by implementing a set of abstract interfaces in terms of functionalities provided by the layers where porting takes place.

In our system, we consider a series of power consuming components that are accounted as the main hardware power consumers of the system. We investigated the following six components: CPU, audio, display, WLAN, Bluetooth and the hard drive keeping the file system. According to their specifics, every component is performing different measurements over an abstract monitoring interface. The monitoring interface decouples the power efficiency framework to any platform specifics and thus increases its portability.

The data monitoring (or data logging) interfaces of the framework provide efficiency measurements based on the component measurements introduced in previous chapters, where the

raw data is collected by a set of platform drivers. Power efficiency measurements can be used to assess the energy efficiency of the platform in terms of different workloads that are applied.

A feedback loop is designed in order to improve the power efficiency dynamically, by enabling applications or the operating systems to take dynamic decisions at runtime. For example, in case of a wireless application, the application might adjust the download rate or the wireless driver might adjust the power of the emitted signal. In case of the CPU adjustments, applications might insert wait states into their algorithm logic or adjust the priority of the worker threads in order to increase power efficiency.

The mechanism works on the principle of high level metrics. The basic idea is that the monitoring framework defines a set of high level and domain specific application metrics where applications are registering their execution threads. The framework is informed by the application whenever a change occurs in the profile configuration data (e.g. download rate is changing). Based on the registered metrics and the available system measurements, the monitoring framework performs active monitoring and provides a feedback loop to applications or the operating system.

The general system overview with the integration of the power efficiency framework is depicted in Fig. 5.

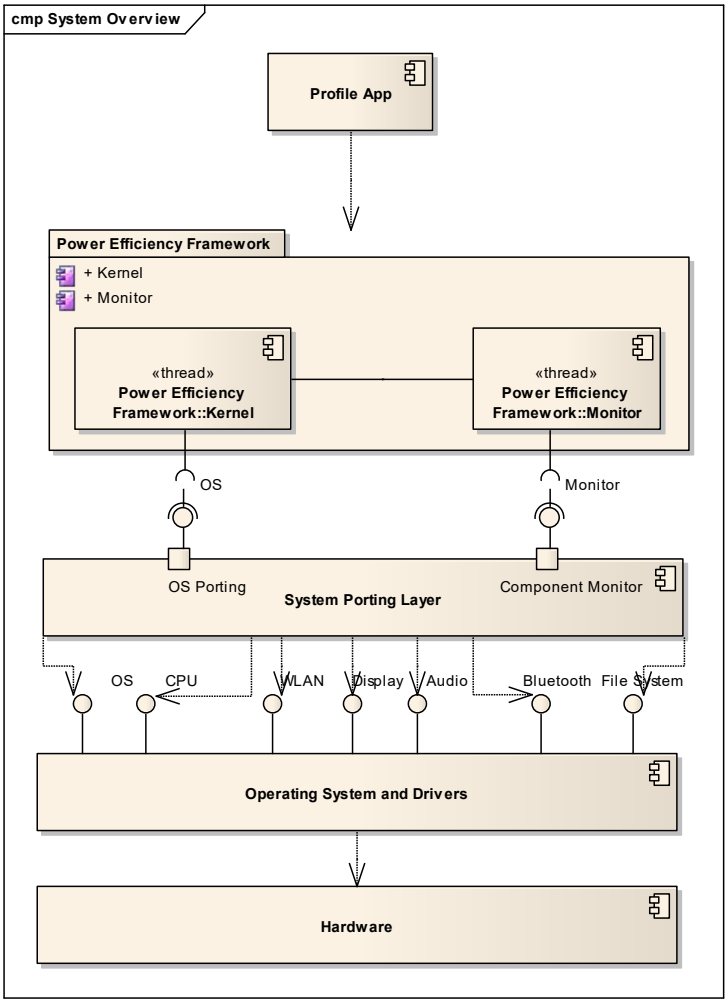

Fig. 5 Power efficiency framework architecture

On the bottom level there is the hardware platform of the system. On top of the hardware platform, there is usually the operating system together with platform drivers which provide the low level functionalities for all the hardware features. On top of the operating system and driver layer there is a system porting layer which glues together the power framework to the runtime executive and drivers. The presence of the porting layer is an effect of requirement that the power efficiency framework should be platform independent.

The porting layer must implement the required Monitor and OS interfaces of the framework in order to provide basic operating system functionalities such as threads, timers and memory management. The Monitor interface is the gateway for collecting measurement data. Monitoring data comes in terms of the specifics of each of the supported power components.

On top of the system porting layer there is the core of this work: the power efficiency framework component. It has two main components: a kernel and a monitor. The kernel provides all the necessary logic and exposes a generic interface towards the higher layers. At the same time, the monitoring interface abstracts the collection of measurement data and their representation and provides it to the kernel as necessary.

On top of the framework we normally have applications which are standard applications which are running on the system. For evaluation and profiling purposes, we supply a set of profiling applications which aim to apply different kinds of stimuli to the system in order to determine the system characteristics in terms of power efficiency. Profiling determines a set of relevant system states, which are determined in isolation (each component is profiled in one step). Each state is associated a power consumption profile which is configured in the system. This information is later used by the framework in order to determine different consumption and efficiency predictions.

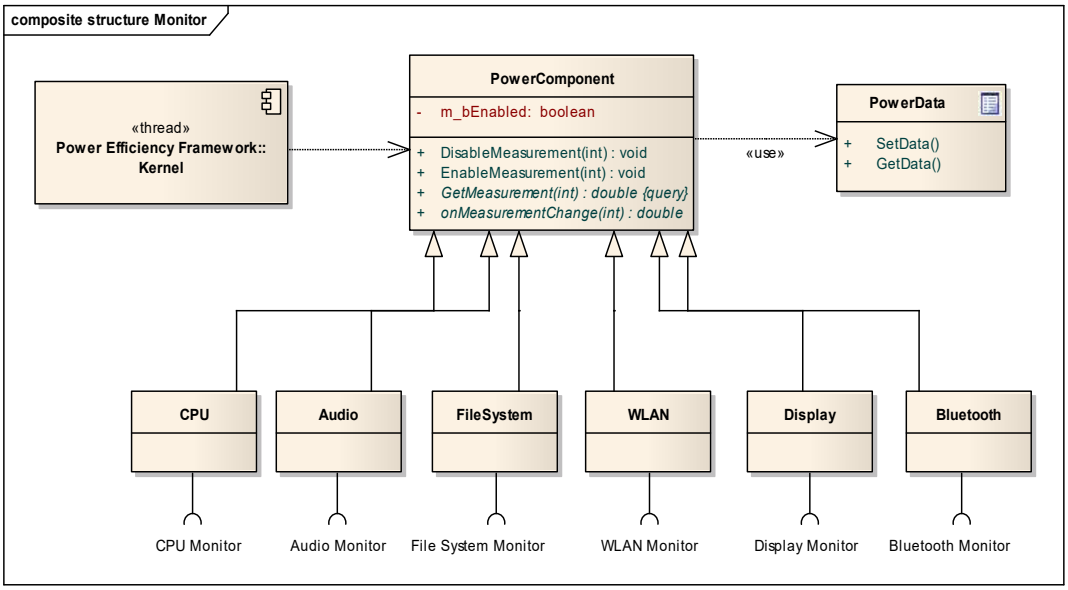

Fig. 6 Power efficiency framework monitor

The details of the Monitor component are shown in Fig. 6. The Monitor component runs in its own execution thread and collects data from different concrete components. Every power component has a set of measurement parameters defined. Power components define a required interface through which different measurements are collected. These interfaces are implemented by the porting layer. The implementation of the porting layer defines if the component is enabled or not in the system, as well as which measurement parameters are provided by the physical platform.

Both polling and notification mechanisms are supported for data collection. Upon data reception, the monitoring component stores the data in its own data store represented by the PowerData component. PowerData storage is not persisted. One important aspect is that the translation between the power efficiency framework's data representation and the one provided by the platform is realized in the porting layer.

The kernel of the power efficiency framework runs in the context of an execution thread and is depicted in Fig. 7. The kernel is responsible to provide to the higher layers power efficiency metrics such as information on power consumption (average, instantaneous), power efficiency and to provide predictions on remaining power under the trend of the observed workload. The kernel is also responsible for the metric split among execution threads and applications so that a global picture of the contribution of each application to the global power consumption is given. The contribution can be also provided per individual component or application per component. The general idea is to construct flexible and meaningful reporting facilities in the kernel.

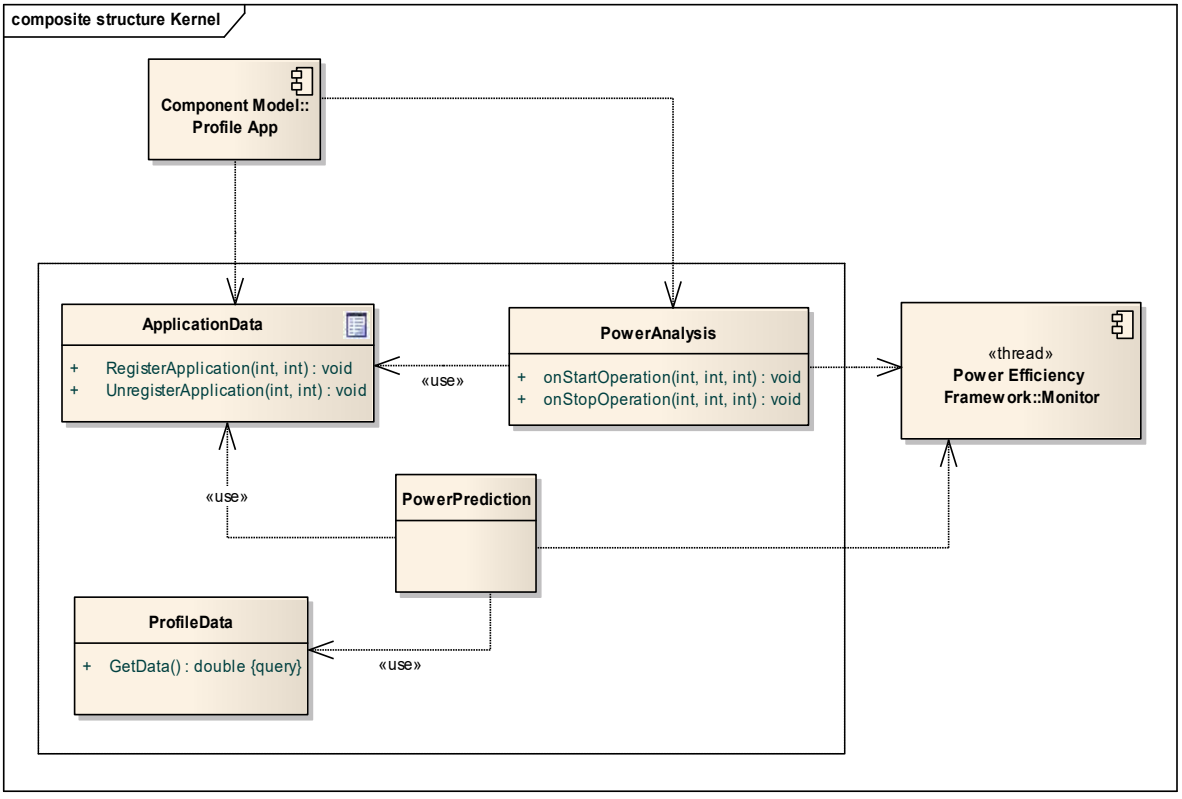

Fig. 7 Power efficiency framework kernel

The client applications register to the kernel via the ApplicationData interface. Basically they specify their set of threads and which components they are interacting with. For example, an application might have a thread which is receiving audio data from a wireless LAN interface and decoding the audio stream (in software) and rendering the stream to the audio driver. The ProfilingData class is a data store which is constructed by the profiling application. The kernel provides a special mode where it collects monitoring data and builds the repository. The profiling data are permanently stored on the framework and can be re-initialized by running the system profiling application again. Profile data is used by the PowerPrediction component to perform prediction on remaining battery capacity or remaining operational time.

The kernel supplies high level metrics on power efficiency through the PowerAnalysis interface. This interface is also used to signal the framework that a given operation has been triggered by an execution thread. This is especially important in cases where the framework cannot make any inferences on the power consumption split among execution threads and components. For example, for power consumption due to audio accesses, all audio accesses to the audio driver need to be instrumented so that the kernel can build their real distribution. If such usage information is not supplied, the power analysis component of the kernel provides a fair split between execution threads and components.

Based on the proposed framework architecture we implemented the framework prototype. During detailed design several tradeoffs and challenges have to be addressed. Here are some of them:

- we intended the power framework to be portable over a number of different hardware and OS architectures, therefore we provided the porting layer in the overall design. Initially this layer was considered as a thin wrapper for the operating system and device drivers services. But if we consider the significant differences between various OS APIs and the large number of components, this layer become quite complex and resource consuming. Here are the solutions to this problem we considered: renounce the porting layer and replicate the framework for every target OS; develop the porting layer no matter its complexity and its time response; design only the interfaces for the needed components (battery, CPU, and WLAN); introduce the concept of abstract components that wraps the hardware component API and decide for every component if it is implemented in the porting layer or not.
- an increasing amount of data has been sampled by monitoring component during execution lifetime of the benchmarks. Considering the limited memory, storage and processing resources available on mobile devices special attention had to be paid when managing these data: keep all measurement data persistent or only the high level metrics, make use of a light database for mobile systems and select the proper way to analyze monitored data. There are different possible solutions to address these problems: use an external database for measurements and extract relevant metrics using data mining techniques; use memory structures for selected measurements and extract metrics using the proposed model.
- threading and synchronization at the framework kernel level implies some overhead due to the context switching and locking operations. There are some possible solutions expose the functions through the porting layer; use directly the OS API in the framework kernel or do not used different threads in the kernel. We decided to use a small number of threads (two

threads) with minimum locking sections and the implementation is based on the OS API, not the porting layer.

- communication between framework and user applications is another design aspect which needed a design decision. There were different solutions investigated. First, the framework continuously monitors the target processes and threads and take decisions based on that; in this case there is no communication between process and framework. The second solution is to register the user power-aware applications into the framework using the available interface and further the applications communicate with the framework using inter-process communication. The last solution makes use again of registration interface but the communication between applications and framework is through function calls. Therefore in the last case the applications are to be compiled and build with the framework.

The proposed solution in Fig. 8 is based on a layered architecture containing the user interface layer (GUI layer), energy-efficiency logic specific layer (Profiling layer), component parameters monitoring (Monitoring layer) and the data acquisition layer (Data layer). The bottom layer is running on top of the host operating system (OS) and it provides a platform independent method to access and collect specific parameters from the underlying hardware and software components. Data acquisition layer uses the specific APIs provided by OS to read component related measurements which are further transferred to the upper layer in a uniform platform independent way. But the implementation of data access layer is platform dependent; therefore specific implementation should be developed for every new targeted platform or device. This layer provides access to read battery, telephony, radio, CPU and memory parameters.

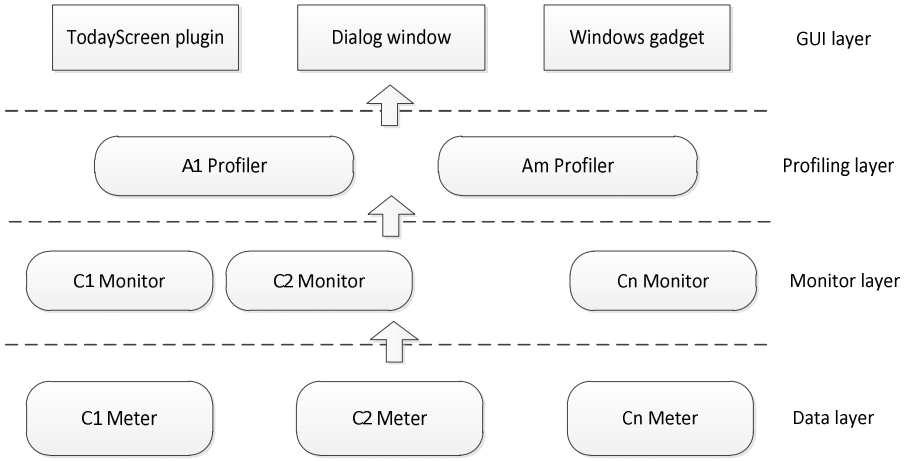

Fig. 8 Energy efficiency framework layers and modules

The middle layer is platform independent and it implements the energy estimation logic. These layers contain intelligent component monitors and the energy-efficiency profilers matching the power consumption measurements and workload related phases to estimate the energy consumed by every specific workload. The monitor components store historical data which are used to calculate energy costs of each workload.

On top of the logic layer is placed the user interface layer which provides mobile users with simple access to energy costs of the last workloads and an estimation of the available time considering

the current energy status of the battery and the current running workloads. The GUI layer is platform dependent and is designed to allow development of application specific GUI components: today screen or normal dialogs.

Component monitoring solution shown in Fig. 9 is based on four design elements: factories, meters, samples and monitors.

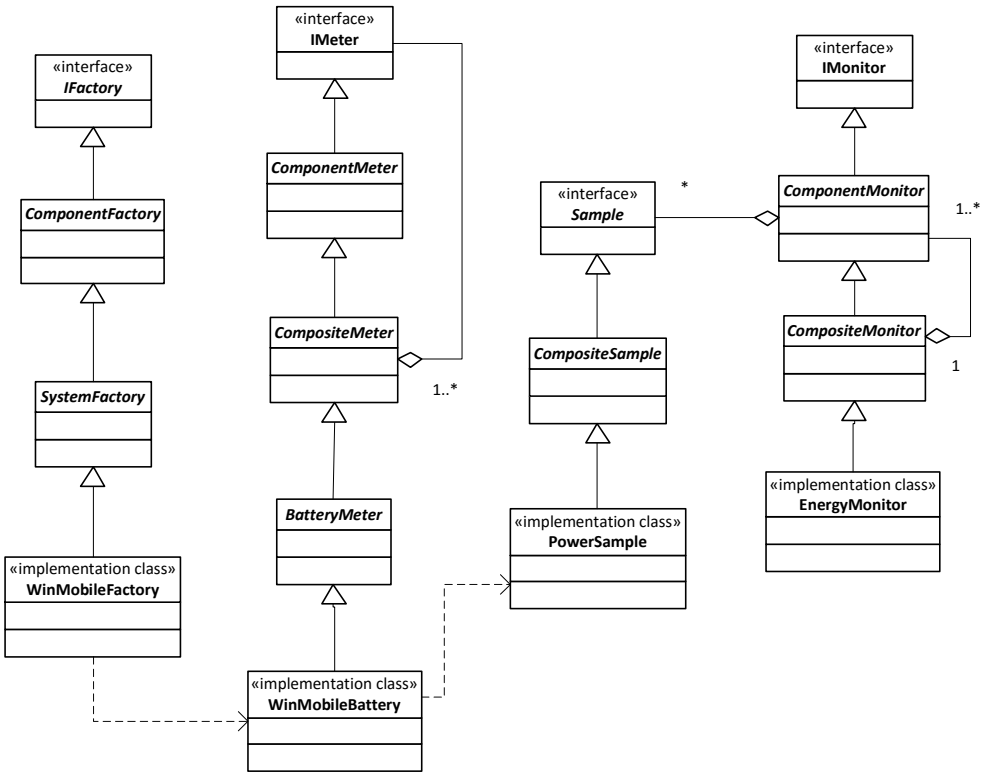

Fig. 9 Energy monitor design

Factory objects are used to identify the host system, to detect the target components in the system and finally to create meter objects for the detected components. Factory objects implementation is platform dependent and they generate platform dependent metering objects. Meter objects are used to provide read access to components in the system or OS to collect components specific parameters. Meter objects implementations are also platform specific. The meter objects can be used by polling the components or by listening from certain events from the component.

Sample objects are platform independent and they are used to store the parameters read from a component for a sampled moment of time. The Sample objects are stamped with time information. Battery parameters are stored in the data type PowerSample which contains: current, voltage, remaining time, capacity, remaining percentage and temperature. The main problem of measuring and using parameters of different batteries is that they are not all available on every battery driver. The devices we used do not provide current or power consumption parameters. In this case we can estimate the consumption based on remaining lifetime, voltages and remaining percentage.

The last element type in the monitoring design is the monitor itself. The monitors are platform independent and they have specific implementation for every type of observed component. The sampled data from the meters are temporary stored in these monitors. The monitors extract relevant information from historical parameters in order to try to find models, to extract patterns or to predict trends.

The upper layer containing aggregation components collects component specific metrics from different monitors and computes the application specific energy efficiency values. For example at the aggregation level, one module can compute the energy consumption or energy efficiency of a specific workload. Based on these computations, online energy measurement of mobile applications can be achieved.

2.4 COMPONENT PROFILING

Hardware component's power consumption depends on runtime workload this component executes which in turn is specified by the applications running by the device. The overall power consumption of the whole system is composed of power consumption of every component. Sometimes reducing power consumption of one component imply and increase in consumption of other components, therefore the system as a whole should be also considered in the proposed framework. The power framework is the central part for the running applications which have to balance and optimize the applications' power management decisions in order to obtain overall reduction in power consumption or energy.

Both workload and power profiles of a physical device are based on two elements: the first one is the set of power states the device may have and the second one is the set of different active usage patterns or workloads available for every power state. Every runtime usage pattern for a certain power state has its own power consumption level, in other words if a component executes the same operation in the same conditions it will consume the same power level.

The power profiles we defined establish a relationship between power states, workloads and power consumed by the device or component being in these states. In our framework we characterize power states having certain power consumption levels, in terms of the effects observed on the battery' characteristics. These profiles have to be achieved empirically by the mean of the power profiling tests.

Conceptually, these profiles can be grouped in power consumption classes based on two kinds of parameters: system (usage) parameters and battery (consumption) parameters. Fig. 10 describes three types of power consumption profiles for the following measurements: consumed energy, remaining energy and dissipated energy. Every profile is dependent on the workload executed by the system. The workload profile determines the power states the device can switch while running (Fig. 10.a) and thus it determines the power consumption profile of the device. Power consumption shown in Fig. 10.b has different power consumption values for a component at different workload type. Every workload type has associated a certain power consumption level. Power consumption profiles are usually described as a sequence of power consumption levels associated with a sequence of workload type.

Remaining energy, which is shown in Fig. 10.c, is the energy available in the device energy supply (usually the battery) that can be further used by the device. The remaining energy is decreasing with time depending of workload profile. The available energy is decreasing faster when higher workload levels are applied to the component. Dissipated energy is the fraction of the consumed energy loosed by the system as heat (Fig. 10.d).

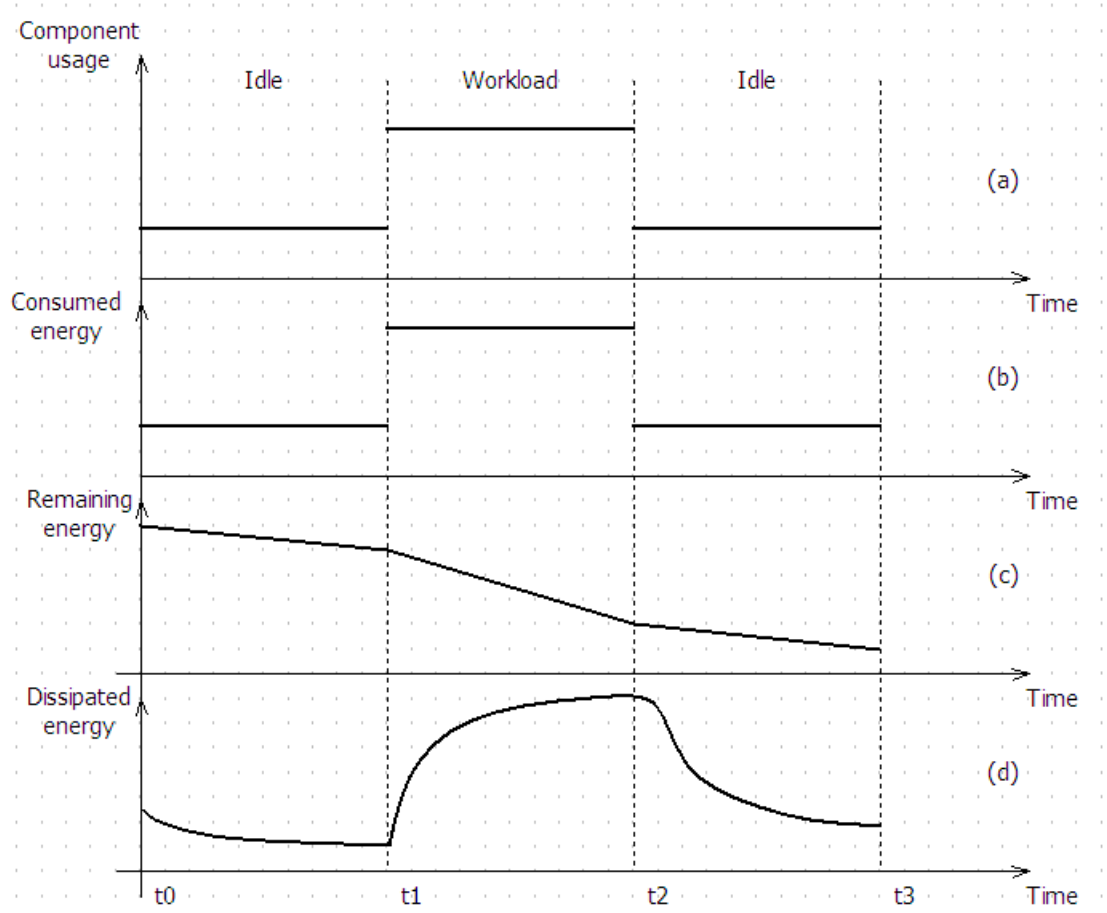

Fig. 10 Theoretical power consumption profiles

To conclude, the power efficiency framework is based on following concepts:

- Workload components – are considered the system's physical modules that consume power and contribute to the overall power consumption of the system. For a mobile device we identified the following components: CPU and memory, WLAN, Flash file system, Audio chipset, Display, Video chipset, Bluetooth, GPS receiver, GSM chipset, etc.
- Component measurements – define the measured parameters available for every addressed component. For every component in system there are a set of measurable parameters which describe how the component is used and its power consumption state. Every component parameter is included in one of the four classes presented in Fig. 10. For the same component, different systems used it may provide different parameters, depending on the component driver implementation. Measurements could be enabled/ disabled through configuration files depending on the system.

- Power efficiency metrics – are defined as high-level metrics computed by the power framework core, which makes them available to user applications through a well-defined interface layer. High-level metrics are domain specific values computed for every component and registered user or system application. These values are domain specific in terms of power consumption for component or application workload operations. Some basic component measurements, like power consumption metrics, could be propagated from lower level of the framework to user applications. Power consumption and power efficiency metrics could be computed as average value or instantaneous values. Beside power efficiency metrics computation the framework core has to split these values and dispatch them in order to obtain:
    - contribution of individual components to the global power consumption
    - contribution of each application to the power consumption per component
    - contribution of each application to the global power consumption
- Power profiles – specify how power efficiency metrics varies in time. We define one set of profiles per component in order to see how component power consumption changes with component's parameters. Power profiles establish a relationship between power states, workloads and power consumed by the device or component being in these states. Profiling determines a set of relevant states of the system which are determined in isolation (no other components are enabled). Each state has associated a power consumption profile (e.g. amount of used energy per activity). Based on component power profiles we intend to obtain power consumption forecasts for certain workloads before they are executed.

2.4.1 Physical measurements

Typically, power related measurements correspond to the main system power sources: CPU and memory, WLAN/LAN card, storage system, display, video and audio. Every system component has a number of parameters provided by its driver or by the operating system interfaces. In this section we address the first two components: the CPU and memory as a single component because for the higher levels they are tight related. These parameters are presented in Tab. 1.

Item	Physical	Operating System	Application
Battery	I [mA] - current C [mWh] - battery capacity V [V] - battery voltage P [mW] - discharge rate T - temperature	Power states	Workload type
CPU	T - temperature F - fan speed Ps - p-state	CPU load CPU counters Memory usage Thread info	Thread count

WLAN	RSSI - power strength Security model		Bandwidth Error rate
Display	Brightness level		Brightness level
Audio	Volume level		Volume level

Tab. 1 System and component measurements

We grouped the measurements in four classes (Fig. 10):

- Workload sources measurements - specifies the description of the components which may consume energy in the system.
- Battery energy measurements - specifies the current status of the remaining energy available in the system.
- Power consumption measurements - specifies the current system power consumption
- Heat dissipation measurements - specifies the energy lost in the system through heat dissipation.

The main goal of our tests is to find an empirical relation between the extracted measurements for the CPU component and its power consumption. In order to achieve this relation we used power benchmarks to profile the system and the applications, which are defined in the next section.

CPU measurements

CPU is the most important component in the system and it has a substantial contribution to the overall systems' power consumption. CPU parameters could be measured globally for all applications running in the system or specifically for every application registered in the framework. There are two types of CPU parameters: static and dynamic. Static CPU parameters are known a priori for a certain system and could be configured through configuration files or can be obtained using OS API. Dynamic CPU parameters could be measured in real time (online) for the running applications and they are variable with the CPU utilization by these applications. Dynamic parameters could be achieved for the whole system or per application, but static CPU parameters are only global parameters. Dynamic parameters are computed as an average on a fixed amount of time which is a settable parameter for the framework, and we call it framework sample time period.

We consider in our model that one application is composed, in terms of CPU, from one or more threads. The application threads could be started when the application is launched (as a pool of threads) and they are alive during the whole application lifetime or the application could start dynamically a number of threads, when they are needed. Every application thread has a unique ID, the processor core it is allocated to, the priority and CPU and memory usage by the thread during the current time period. The CPU power consumption due to a certain application is to be computed from CPU parameters computed for every thread in the observed application.

CPU core parameters are used to characterize statically and dynamically the CPU configuration of the system:

- CPU cores number (cpu_{no}) - specify the number of cores available in the system. This parameter is static and is well defined globally for a certain system. We consider for the current implementation only single-processor systems with single or multiple cores. [cpu_core_number]
- CPU core available power states (cpu_{ps}) - every CPU core has a well-defined number of power states: active, low power, sleep, inactive. For every power state the power consumption level has to be known and statically configured. A CPU power state is defined by its name, power consumption and the activation time needed by the CPU core to enter this power state. [cpu_power_states_number, cpu_power_state_name, cpu_power_state_value, cpu_power_state_uptime]
- CPU core runtime power states - specify in every sample in time the current power states the CPU cores are used in. For every core, the runtime power state is one of the available power states for the CPU presented before. [cpu_runtime_power_state]

CPU usage parameters specify how the CPU is used by all running applications:

- Global CPU time (cpu_{gt}) - specify the time the CPU or its cores are used by the running programs' threads in the current evaluation time period. CPU time is expressed in clock tick counts and defined as the amount of time the CPU is actually executing instructions of the running applications. Global CPU time is the difference between the total amount of time the CPU executes instructions counted from the system begin time to the sampling interval end time and the amount of CPU time from the system begin time and the start time of the sampling interval [global_cpu_time].

Thread usage parameters specify how the CPU cores are shared by the running threads:

- Thread CPU time (cpu_{tt}) - specify the amount of time a thread has actively used the CPU core during the framework sample time period. This parameter is computed for every combination of thread and CPU core in order to show how a thread was executed and migrated on different cores. The same with global CPU time, it is expressed in clock tick counts and can be computed by difference between the total amount of time the thread used the CPU from its begin to the stop time of the current sampling interval and the amount of CPU time from the thread begin and the start time of the next sampling interval [thread_cpu_time].

Wireless measurements

WLAN interface has an important contribution to the total system's power consumption, especially for smart mobile devices and tablets. Typically, the main contributors to power consumption of wireless interface are the chipset and the radio interfaces. They are under the control of the applications that are performing wireless communication and OS drivers implementing the protocol stacks.

The power consumption measurements used to profile wireless communication:

- WiFi chipset power consumption ($P_{wifi-on}$) - specifies the amount of power consumed by the wireless interface hardware when the interface is switched on but it is not connected to an access point.
- WiFi connected power consumption ($P_{wifi-con}$) - specifies the power consumption of the WiFi circuits when connected to an access point and listening to the radio interface.
- WiFi transmission power consumption ($P_{wifi-tr}$) - specifies the power consumption of the WiFi radio interface while sending data packets. Several parameters are taken into account while measuring transmission power consumption: transport protocol, data block size, payload data patterns, received radio signal strength, and transmission rates.
- WiFi reception power consumption ($P_{wifi-rc}$) - specifies the power consumption of the radio interface while receiving data. The same parameters are taken into account while measuring receiving power consumption: transport protocol, data block size, payload data patterns, received radio signal strength, and reception rates.

Energy efficiency measurements of wireless communication:

- WiFi transmission energy efficiency ($E_{wifi-tr}$) - represents energy consumed by the wireless interface to transmit 1 bit of information under certain conditions: transport protocol, data block size, payload data patterns, received radio signal strength, and transmission rates.
- WiFi reception energy efficiency ($E_{wifi-rc}$) - represents energy consumed by the wireless interface to receive 1 bit of information under certain conditions: transport protocol, data block size, payload data patterns, received radio signal strength, and reception rates.

Battery measurements

Battery parameters are valuable metrics for improving mobile systems and applications power management. Power management strategies applied for lower levels in the system usually do consider neither battery parameters nor other system parameter; they only implement certain policies, in most of the cases based on timeout or usage predictions. Mobile systems are subject to different and complex user and usage models, each with its own battery life profiles. Higher system level and finally the user applications should be aware of their impact on overall power consumption using the set of low level metrics provided by devices, drivers or operating system components. Besides battery parameters there are other operating system or device level parameters which can be used in the power-aware framework.

Our proposed power execution framework is based mainly on the information measured or estimated by the battery driver or the operating system. Battery status is based on several parameters: battery capacity [mWh], maximum battery capacity [mWh], charge/discharge rate [mW], current drawing [mA], battery remaining life time [s], battery temperature [oC], etc. These parameters depend on the way the battery discharges and the power profile of the workload sources.

As a battery discharges, its terminal voltage decreases from an initial voltage V_{max}, given by the open circuit voltage of the fully charged battery, to a cut-off voltage V_{cut}, the voltage at which the battery is considered discharged. Because of the chemical reactions within the cells, the capacity of a battery depends on the discharge conditions such as the magnitude of the current, the duration

of the current, the allowable terminal voltage of the battery, temperature, and other factors. The efficiency of a battery is different at different discharge rates. When discharging at low rate, the battery's energy is delivered more efficiently than at higher discharge rates. Battery *lifetime* under a certain load is given by the time taken for the battery terminal voltage to reach the cut-off voltage. The *standard capacity* of a battery is the charge that can be extracted from a battery when discharged under standard load conditions. The *available capacity* of a battery is the amount of charge that the battery delivers under an arbitrary load, and is usually used (along with battery life) as a metric for measuring battery efficiency.

Many chip vendors offer their own energy consumption specifications on product data sheets that are difficult to compare with one another. When using these circuits in the design phase the attempts to compare processors integrated into system-on-chip implementations, interpreting these values becomes even more difficult. Vendors also use typical power metrics to characterize their processors, but only rarely they indicate the workload that was applied while making these measurements and this will have a significant impact on energy and power numbers.

Modern operating systems provide a comprehensive and system-wide set of power management features in order to extend battery life and save energy, reduce heat and noise, and help ensure data reliability. The power management functions and messages allow applications to retrieve the system power status, notify applications of power management events, and notify the system of each application's power requirements. Intelligent batteries which provide power for recent portable computers and mobile devices are available to applications through the operating system, which provides information on the state of the battery so applications can provide useful functions for the user. To obtain information about the battery status, the applications may use the OS functions, which return general information about all power sources in the system and some estimations of the battery current status computed by operating system: battery life percent, battery life time and battery full life time. In our tests these data are not sufficient for a good estimation of power consumption; therefore detailed information about each individual battery is necessary. In order to obtain detailed status parameters on batteries, the battery drivers should be used.

Temperature measurements

Temperature in computing systems is a valuable parameter which can provide different information about the usage and healthiness of the system and its components. When temperature of a component increases over a certain level, the component is overloaded or could have some hardware defects. On the other hand, the temperature of a component gives an indication of the energy dissipated by the component through heat. Another aspect is related with the increasing of power consumption of a component where temperature of the component increases. Some components in a computing system have already integrated built-in temperature sensors which can be read from applications:

- CPU temperature and core temperature
- Battery temperature

2.4.2 Power benchmarks

The power framework continuously monitors the components and systems' parameters and computes power efficiency metrics which are provided to the registered applications. Every power source in the system has a power profiler in order to estimate and predict its power consumption variation during its usage. We understand by power profiles the variation in time of power efficiency metrics related to the workload applied to the component. In order to use power profiles we need to introduce a standard way to generate the profiles and in order to identify correctly power variation of the components. This standard method to generate power efficiency profiles we called power efficiency benchmarks that we describe in the following section. Besides the two entities we introduced before (power consumption metrics are the high level values we compute based on measurements and power consumption profiles are the way power metrics varies in time for certain workloads) we introduce the power benchmarks which are standard method proposed to extract power profiles.

We propose to extend the concept of benchmarking with another metric: the power consumption and name it power benchmark.

Definition - Power benchmark is defined as a software program that characterizes the power consumption of a system, component or application with respect to certain stimulus (workload).

The metrics used by power benchmark software are based on the measured or estimated power consumption by the battery driver or the operating system.

Many chip vendors offer their own energy consumption specifications on product data sheets that are difficult to compare with one another. When using these circuits in the design phase the attempts to compare processors integrated into system-on-chip implementations, interpreting these values becomes even more difficult. Vendors also use typical power metrics to characterize their processors, but only rarely do they indicate the workload that was applied while making these measurements and this will have a significant impact on energy and power numbers. Therefore an online power benchmarking tool is needed in order to compare all these metrics with respect to real-life scenarios and applications.

A power benchmark must be able to distinguish the way power consumption is increasing with the workload compared to the idle state power consumption. The power benchmark is able to identify the type of workload out of the power consumption variation. Therefore, we define a power benchmark as having three intervals (Fig. 11):

- The first time range $[0\text{-}t_1)$, is intended for idle mode power consumption. In this step, the component does not execute anything, but the power saving mechanisms is prevented to occur.
- The second time range $[t_1\text{-}t_2)$ represents the workload phase, when a certain stimulus is executed. SPEC CPU2000 or any type of other applications can be executed as workload.
- The last time range $[t_2\text{-}t_3)$ represents the releasing phase intended for the component to reach again the idle state power consumption. In this step, the component does not execute anything, but the power saving mechanisms is also prevented to occur.

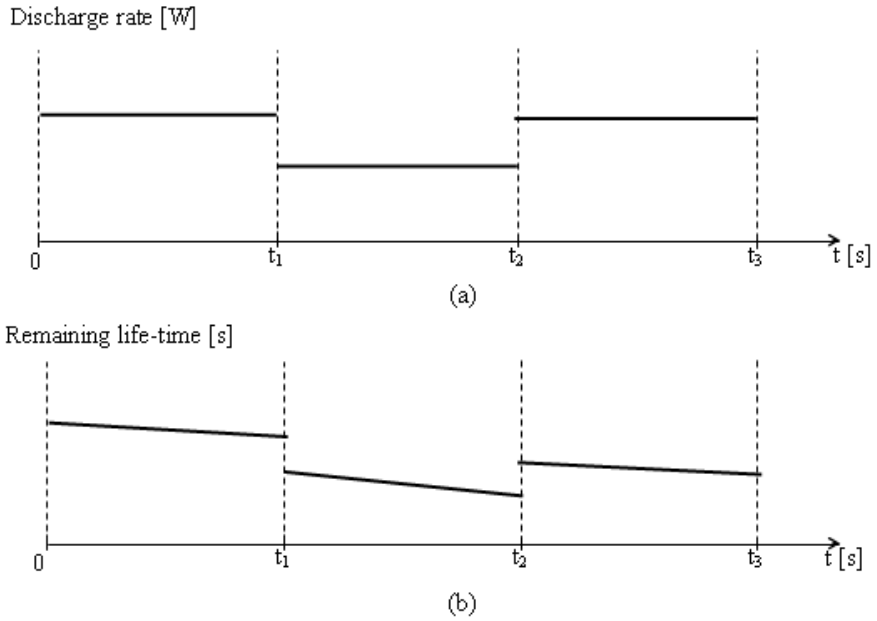

Fig. 11 Power benchmark definitions

There are two ways the power benchmark can be implemented:

- Fixed times power benchmark – the benchmark moments t_1, t_2 and t_3 are predefined and constant. This kind of power benchmark shows the maximum component power consumption when a certain workload is applied for a constant period of time (t_2-t_1);
- Fixed power consumption values power benchmark – the benchmark times are variable and benchmark power measured values are predefined and constant. This benchmark shows how many workload operations are executed per consumed energy.

The power consumption measurements presented before can be used in benchmark, and these can be grouped in two classes: power consumption measurements – (a) (discharge rate, current) and remaining battery capacity measurements – (b) (remaining time, remaining capacity, etc.).

2.4.3 Power efficiency metrics

The kernel is responsible to provide to the higher layers power efficiency metrics such as information on power consumption (average, instantaneous), application or component power efficiency and to provide predictions on remaining power under the trend of the observed workload. The kernel is also responsible for the power metrics split among execution threads and applications so that a global picture of the contribution of each application to the global power consumption is given.

Power efficiency metrics specify how much energy is needed to complete an application specific job. The applications running on top of the framework should be able to estimate the time and the energy needed to finish their tasks in order to adapt and optimize them or at least to establish whether they can be finished in the available energy. Usually applications have the highest chance to estimate the duration of their own tasks, therefore they should know or could estimate how long

a job will last in order to complete. For example an application which downloads a file knows how long will take to complete and could then estimate how much energy is needed. Of course there are also a lot of jobs that cannot be estimated exactly in time but other heuristics or statistical data can be used, but these are application specifics and are not the purpose of our work.

Based on the theoretically concepts introduced before, we define a computing system as a set of hardware components and the software applications using them. The power-aware applications which use the framework need an interface to register them in the framework and an interface for power efficiency data retrieve and analysis. Power analysis interface implemented by the framework provides a way to achieve global power consumption of the system, power consumption of every component in the system or its value from the global power consumption and power consumption of every software application or its relative value from the global power consumption.

Global metrics are the parameters that characterize the power consumption and power efficiency of the whole system:

- Global power consumption (P_s) is the average power consumption of the system during the framework sampling time interval. This metric is the absolute power consumption expressed in Watts and is obtained from the battery measurements presented before. In case the battery driver does not provide any information that could be used to estimate absolute system power consumption, relative values for this metric are accepted. [global_power_consumption]
- Global energy efficiency (E_s) specifies what are the costs in terms of energy in order the system will finish its work. The efficiency is measured in operations per consumed energy and is computed as the useful work executed by the system per quantity of energy consumed to finish the work. The system's operations could be defined in terms of users' operations, component operations or applications operations. [global_energy_efficiency]
- Total available energy (E_t) is the energy that the system's power sources can supply within specific parameters in order the system works properly (works with the expected level of performance or QoS). The energy is expressed in joules or in terms of battery capacity (Wh). This metric is available from battery measurements. [total_available_energy]

Component metrics are the framework computed values that characterize the power consumption and its efficiency of a hardware component:

- Component power consumption (P_{cpu}, P_{wlan}, etc.) is the average power consumption of the component during every sampling interval. This metric specifies the part in the global power consumption due to the addressed component. Component power consumption is a component specific metric which is computed based on the low level measurements for the component. The metric is expressed either in absolute values (recommended) or in relative values (when poor measurements are available). [component_power_consumption (component)].
- Component energy efficiency (E_{cpu}, E_{wlan}) specifies which the costs in terms of energy are in order the component execute a specific amount of work. The efficiency is measured in operations per consumed energy and is computed as the useful work executed by the

component per quantity of energy consumed to finish the work. CPU energy efficiency is the report between CPU times and energy consumed by the component for the time period the CPU time is computed. [component_energy_efficiency]

Application metrics are the framework provided parameters that characterize the power consumption and power efficiency of a software application:

- Application global power consumption (P_{app}) specifies how much of the system power consumption is due to the specific application. This metric provided by the framework core is computed based on the application's threads measurements and the used components' parameters. The metric is expressed either in absolute values (recommended) or in relative values (when poor measurements are available). [application_power_consumption (application)]
- Application energy efficiency (E_{app}) specifies which the costs in terms of energy are in order the specific application executes its work. The efficiency is measured in operations per consumed energy and is computed as the useful work executed by the application per quantity of energy consumed to finish the work. The efficiency is the report between energy consumed by an application and the application specific parameters, for example energy used per frame displayed by a movie player. [application_energy_efficiency]
- Application component power consumption ($P_{app\text{-}cpu}$) is the power consumption of a hardware component induced by a specific application. This metric is computed by the framework function of the application's threads measurements and specific component parameters. The metric is expressed either in absolute values (recommended) or in relative values (when poor measurements are available). [application_power_consumption (application, component)]
- Application component energy efficiency ($E_{app\text{-}cpu}$) specifies which the costs in terms of energy are in order the component execute a specific amount of work for a certain application. The efficiency is measured in operations per consumed energy and is computed as the useful work executed by the component for an application per quantity of energy consumed to finish the work. CPU energy efficiency is the report between CPU times and energy consumed by the component for an application for the time period the CPU time is computed. [application_energy_efficiency]

2.5 EXPERIMENTAL RESULTS

2.5.1 Mobile voice communication

Voice communication remains one of the main functions of mobile devices even though new features and applications are available. Currently there are a lot of voice communications technologies running on different types of mobile devices, such as VoIP technologies provided by applications like Skype, Viber). The multitude and complexity of devices that implement a large spectrum of multimedia and wireless communication protocols require closer evaluation and understanding in respect to their energy efficiency. In this activity we tried to identify, model and estimate the energy consumption of voice communications on mobile devices based on the energy

efficiency framework and energy profiles. We used TAPI (Telephony API) to identify ongoing and incoming calls and next to estimate their energy consumption.

The main goal of our work is to measure or at least to estimate the energy consumed by a mobile device while communicating in GSM 3G and IEEE 802.11 networks. The mobile device can initiate a call (hereinafter referred to as outgoing call) or it can receive a call (hereinafter referred to as incoming call). During voice GSM calls the mobile device consumes more power than if the device is idle or it does not communicate. In order to estimate the energy of GSM voice calls the first step is to monitor the device components using the operating system's API. There are three elements we proposed to monitor in our tests: the battery status, the call status and the cell tower information. Further in this section we describe the architecture and the detailed design of the monitoring application based on the concepts and framework architecture described before.

Considering the wide range of existing mobile devices, the main requirement for the voice communication energy monitoring application was to design and develop a solution which is portable and extensible. The monitoring solution design should be easily portable on different type of battery powered devices (smartphones, pocket pc, tablet pc, laptop), operating systems (Windows Mobile/Phone, Win32/64, Android) or development environment and language (C++, C#, Java). Furthermore, the proposed solution should be able to collect and to abstract the various types of parameters available on different devices (e.g. different battery drivers may provide different types of battery measures: voltage, current, power consumption, remaining lifetime, etc.)

The monitoring application was tested on two mobile devices: HTC HD2 Windows Mobile 6.5 (WM 6.5) smartphone and Fujitsu Siemens Pocket Loox T830 Windows Mobile 5.5 (WM 5.5) smartphone due to their API access to the smart battery drivers and their measurements accuracy. The monitoring application can be compiled for WM 5, WM 6, WM 6.5 and Win32 for ARM and x86 microprocessor architectures.

The proposed general solution presented in Fig. 5 Power efficiency framework architectureFig. 5 is customized for this specific application which is shown in Fig. 12. It is based on a layered architecture containing the user interface layer (GUI layer), call energy logic specific layer (Logic layer) and the data acquisition layer (Data layer). The bottom layer is running on top of the host operating system (OS) and it provides a platform independent method to access and collect specific parameters from the underlying hardware and software components. Data acquisition layer uses the specific APIs provided by OS to read component related measurements which are further transferred to the upper layer in a uniform platform independent way. But the implementation of data access layer is platform dependent; therefore specific implementation should be developed for every new targeted platform or device. This layer provides access to read battery, telephony and radio related parameters.

The middle layer is platform independent and it implements the energy estimation logic. This layer contains intelligent battery and call monitors and the call energy profiler that matches the power consumption measurements and call related phases to estimate the energy consumed by every voice call. The monitor components store historical data which are used to calculate energy costs of each call.

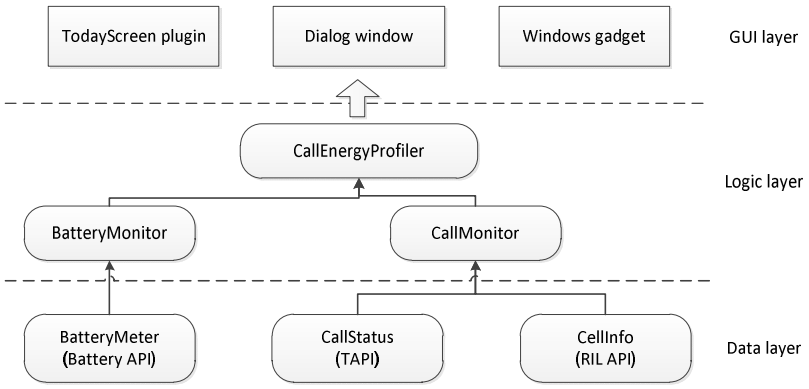

Fig. 12 Overall energy-aware voice solution architecture

On top of the logic layer is placed the user interface layer which provides mobile users with simple access to energy costs of the last calls and an estimation of the available conversation time considering the current energy status of the battery. The GUI layer is platform dependent and is designed to allow development of application specific GUI components: today screen or normal dialogs.

Factory objects are used to identify the host system, to detect the target components in the system and finally to create meter objects for the detected components. Factory objects implementation is platform dependent and they generate platform dependent metering objects. Meter objects are used to provide read access to components in the system or OS to collect components specific parameters. Meter objects implementations are also platform specific. In Windows Mobile OS we based the implementation of WinMobileBattery meter on function GetSystemPowerStatusEx2. The meter objects can be used by polling the components or by listening from certain events from the component. WinMobileBattery is used in polling mode to sample with certain time rate (usually 1 second) the battery parameters. Every sampled set of parameters are stored in Sample objects.

Sample objects are platform independent and they are used to store the parameters read from a component for a sampled moment of time. The Sample objects are stamped with time information. Battery parameters are stored in the data type PowerSample which contains: current, voltage, remaining time, capacity, remaining percentage and temperature.

The last element type in the monitoring design is the monitor itself. The monitors are platform independent and they have specific implementation for every type of observed component. The sampled data from the meters are temporary stored in these monitors. The monitors extract relevant information from historical parameters in order to try to find models, to extract patterns or to predict trends.

Some of the monitoring and metering objects are composite because more than one component of the same type may exist in the system. One complex sample system detected and constructed by a Factory object is shown in Fig. 13. It is a system with two batteries and two SIM cards which needs

two monitor objects, one for batteries and one for radio interfaces, and four meter objects, one for every battery and radio interface in the system.

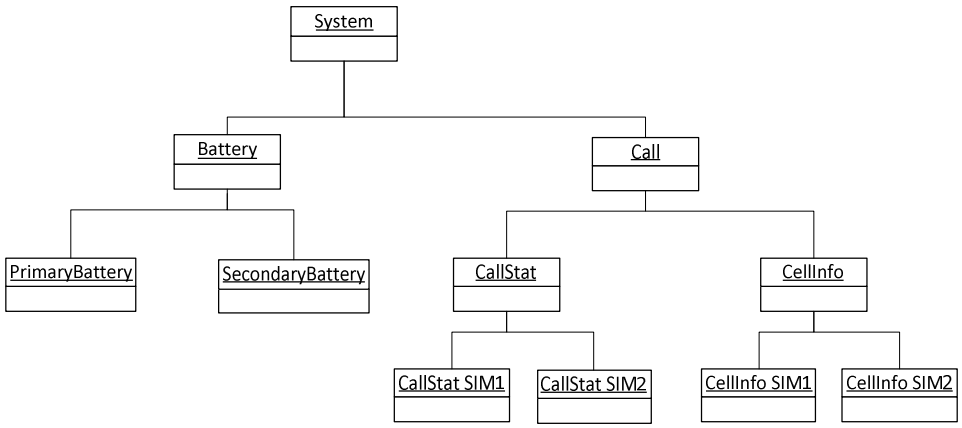

Fig. 13 Metering objects hierarchy

To monitor call related parameters we need two types of information: (1) call status to detect the beginning and the end of an incoming or outgoing call and (2) call information to obtain the radio signal strength as estimation for the distance between mobile station and the Base Transceiver Station (BTS). Call status on Windows Mobile devices can be captured using Telephony API (TAPI). TAPI is used in a Windows environment to monitor and control telephony resources. Cell tower information is accessed using Radio Interface Layer API.

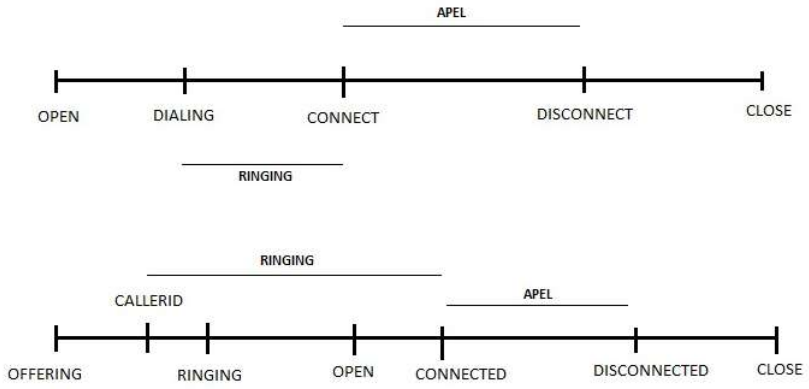

Fig. 14 Outgoing and incoming calls identification

TAPI is an event based programming interface for applications to access, monitor and control telephony services. An event based TAPICallStatusMeter is implemented in the application to detect call related events. The event sequences for outgoing and incoming calls are presented in Fig. 14. However for our purpose only few of the messages are needed in order to identify the beginning and the end of an incoming or outgoing call.

Every voice call on a mobile device implies several states: dialing, ringing, busy, connected, disconnected, and idle. In our tests we considered two call states for energy evaluation, ringing

and connected (Fig. 14), and we compared them with the idle state power consumption of the device. The outgoing call is identified by several relevant TAPI messages:

- LINEDEVSTATE_OPEN generated when the voice line has been opened by the calling application
- LINECALLSTATE_DIALING - the mobile device is dialing the phone number digits
- LINECALLSTATE_CONNECTED - the call has been established and the connection is established between mobile device and the called device
- LINECALLSTATE_DISCONNECTED - the remote party has disconnected from the call
- LINEDEVSTATE_CLOSE - the line has been closed by the calling application

The ringing state of an outgoing call is considered the time period between LINECALLSTATE_DIALING and LINECALLSTATE_CONNECTED. The voice communication of an outgoing call is considered the time interval between LINECALLSTATE_CONNECTED and LINECALLSTATE_DISCONNECTED.

The incoming call is identified by several relevant TAPI messages:

- LINECALLSTATE_OFFERING - the signaling for a new call was received by the device
- LINEDEVSTATE_RINGING - the mobile device is ringing to announce the user the new arrived call
- LINEDEVSTATE_OPEN - the voice line has been opened by the calling application
- LINECALLSTATE_CONNECTED - the call has been established and the connection is established between mobile device and the calling device
- LINECALLSTATE_DISCONNECTED - the remote party has disconnected from the call
- LINEDEVSTATE_CLOSE - the line has been closed by the calling application

The ringing state of an incoming call is considered the time period between LINEDEVSTATE_RINGING (or LINECALLINFOSTATE_CALLERID) and LINECALLSTATE_CONNECTED. The voice communication of an incoming call is considered the time interval between LINECALLSTATE_CONNECTED and LINECALLSTATE_DISCONNECTED.

We executed the call monitoring application on two Windows mobile devices (HTC HD2 and FSC Loox T830). We ran several call tests between devices while monitoring system parameters and power consumption of devices.

Fig. 15 shows power consumption variation of the HTC device during outgoing and incoming calls. An increase of approximatively 1 W can be observed during calls comparing with the idle state power consumption. There is no significant and consistent difference between power consumption of incoming and outgoing calls. Fig. 16 shows power consumption of the FSC device during voice calls. The increase of power consumption during calls is higher than the one of the other device, with 2W higher than the idle state power consumption. However, this difference is due to the older version of the hardware of FSC device.

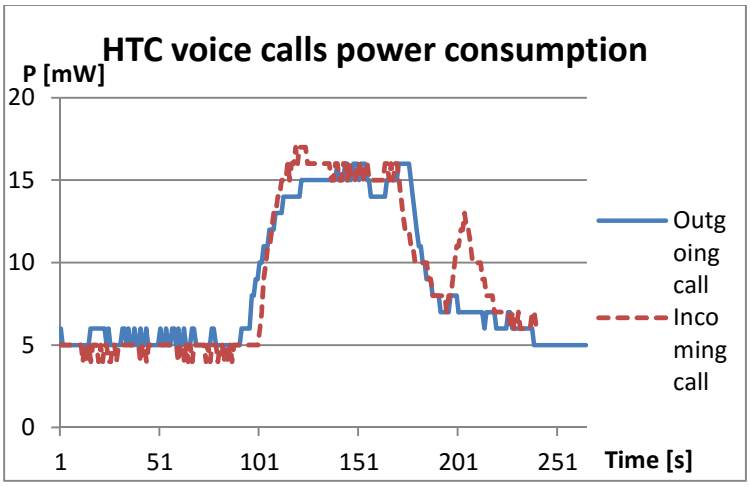

Fig. 15 HTC voice calls power consumption

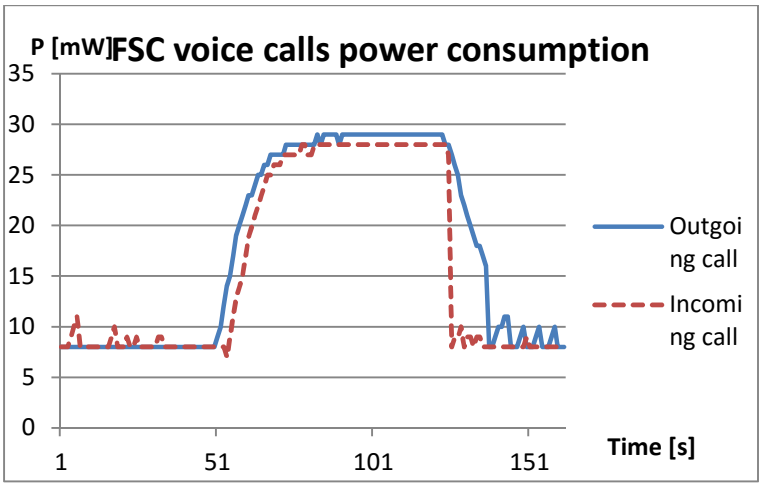

Fig. 16 FSC voice calls power consumption

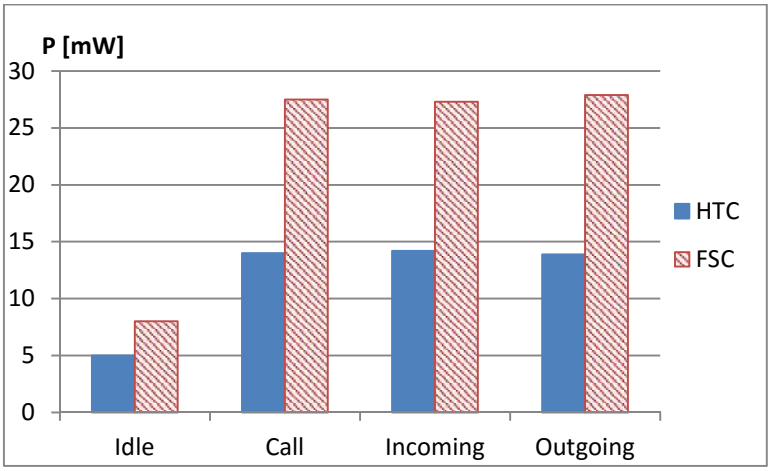

Fig. 17 Idle vs. voice calls power consumption

Average power consumption values for both devices are presented in Fig. 17. In our tests we did not observe significant and repeatable differences between incoming and outgoing calls. However a 100%-200% increase in power consumption is accounted when communicating compared with the idle state power consumption of the devices.

The user interface for Windows Mobile is shown in Fig. 18. The application measure and display in the Today plugin the power consumption of the system when idle and when the user initiates or accepts a call.

Fig. 18 Idle vs. voice calls power consumption (GUI)

2.5.2 Wireless data communication

In order to obtain always connected mobile applications, energy-efficient wireless communication chipsets, protocols, clients and servers have to be developed. With the proposed tests we aimed to highlight the relation between different wireless communication parameters and their effect on system power consumption in order to capture these relations in the final model implementation.

The proposed test set contains the following test cases:

- WLAN power states - to measure power consumption of the device when wireless interface is switched between its available power states: OFF, ON (without communication and without power management), ON (without communication and with PM), ON (with communication in progress)
- WLAN security - to measure power consumption of the device when its WLAN interface is using different security settings and encryption algorithms: OFF, ON (WEP), ON (WPA-TKIP), ON (WPA-AES)
- WLAN TCP/IP transport protocols – UDP and TCP transport protocols have been used

- WLAN communication data patterns - to measure power consumption of the device when transferring different types of data: text, zip, multimedia
- WLAN communication direction - to measure power consumption of uplink and downlink data transfers
- WLAN transfer rates - to measure power consumption and energy efficiency when different transfer rates are obtained
- WLAN radio signal strength and distance between communicating nodes and how they influence the device power consumption

Based on the results of these tests we further tried to introduce them into the proposed energy estimation model. High-level profiling of wireless communication conducts to several conclusions. Power consumption of WLAN power and communication states can be measured and show constant values for every specific device. When transferring data on wireless interface, the power consumption depends on the security model used but it do not depends on distance between emitter and receiver and also it do not depends on data patterns or types (Fig. 19).

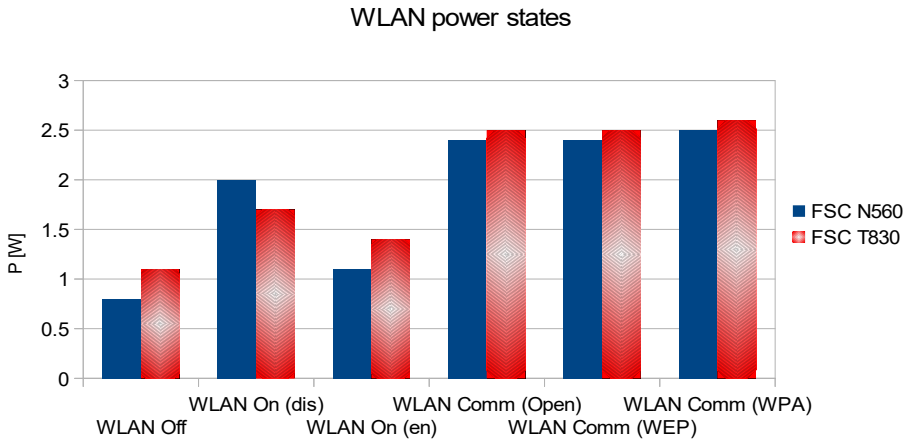

Fig. 19 Wireless LAN interface high-level power states

Power consumption of wireless interface while transferring data is considered constant within one high-level power state (Fig. 20). However, the energy efficiency of communication in one power state is not constant and depends on the achieved transfer rate (Fig. 21). Therefore, WLAN communication is modeled using the two parameters: current power state and transfer rate.

Based on these parameters an energy-aware mobile download manager application aware of its energy consumption can be developed.

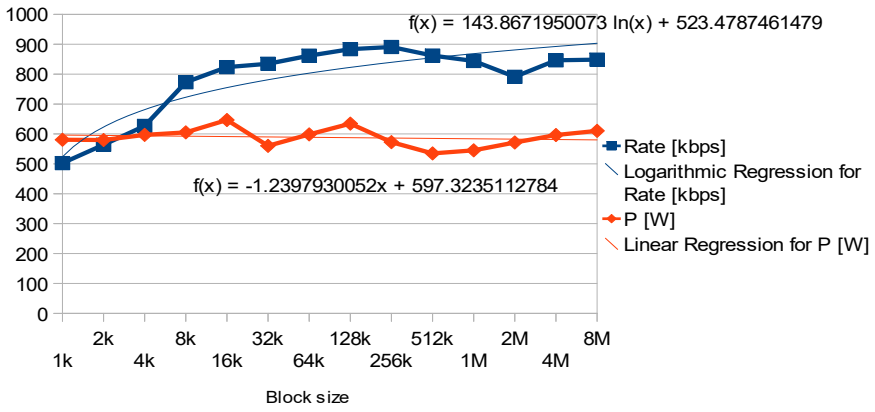

Fig. 20 Wireless interface power consumption and rate

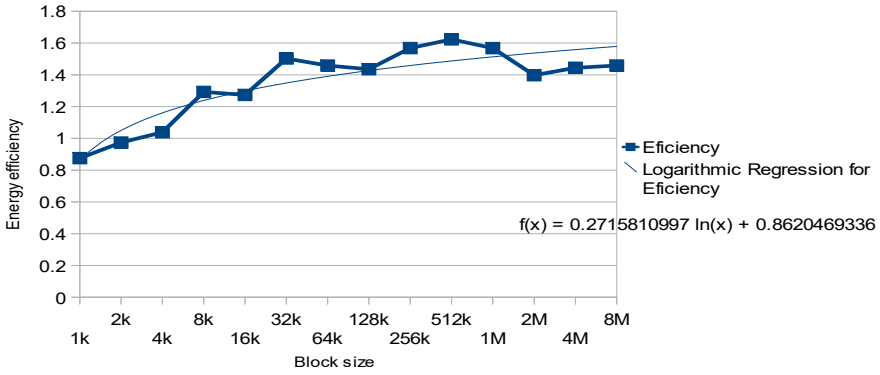

Fig. 21 Wireless LAN interface high-level energy efficiency

2.5.3 CPU intensive processing

Mobile applications follow an ascending trend of processing performance requirements, trend that leads to increasing power consumption values of the processor and to higher temperatures of the CPU cores. CPU intensive workloads are therefore worth to be studied and optimized based on their energy efficiency. An energy-aware application was developed on top of the energy-efficiency framework which implements several CPU benchmark algorithms. An important aspect of CPU intensive workloads is their thermal effect; therefore we have been interested also in temperature profiles of these workloads.

Thermal profile of a processor when executing high intensive computations is shown in Fig. 23. The CPU load is 100% and CPU temperature increases in time until heat dissipation steady state is achieved. When the workload is executed on a multicore architecture the single threaded benchmark application is continuously switched from one core to another on the basis of OS task scheduler implementation. The overall CPU usage for the single threaded integer test application is 50% and the application receives different processor times from every core (Fig. 22). Thermal profiles of CPU and CPU cores, when the same integer workload is executed, are presented in Fig. 23. The CPU cores are not at the same temperature because their uses are not balanced equally.

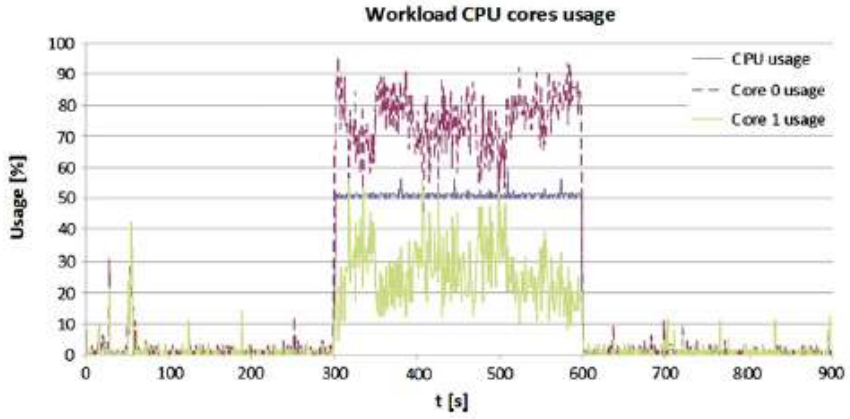

Fig. 22 CPU cores usage level for single threaded test

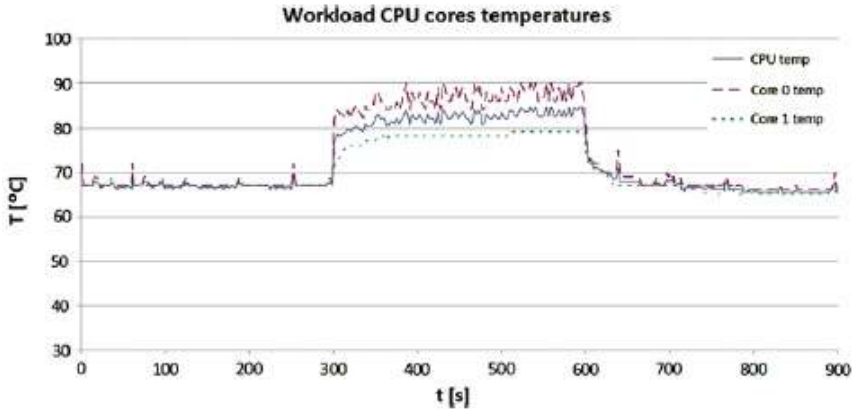

Fig. 23 CPU cores temperatures for single threaded test

In order to implement energy-aware CPU workloads we executed the same workload at different CPU usage levels. The same CPU intensive workload when executed at different CPU loads produces different thermal profiles and power consumption levels. For example the same workload was implemented in three ways:

- full performance (CPU usage=100%) - Fig. 24, the workload is implemented for best performance.
- short delays (CPU usage=50%) - Fig. 24, inside the workload algorithm, after a number of iterations, short intervals sleep function calls have been introduced. Depending on the frequency of sleep calls we could achieve different CPU usage levels.
- long delays (CPU usage=50–75%) - Fig. 25, inside the workload algorithm, with a selectable frequency, long delays are introduced. Depending on the delay periods and frequencies different CPU loads could be achieved, but we also obtained different thermal profiles. Therefore, in some cases we can control and adapt a software application in order to reduce the heat dissipation due to this application (Fig. 25).

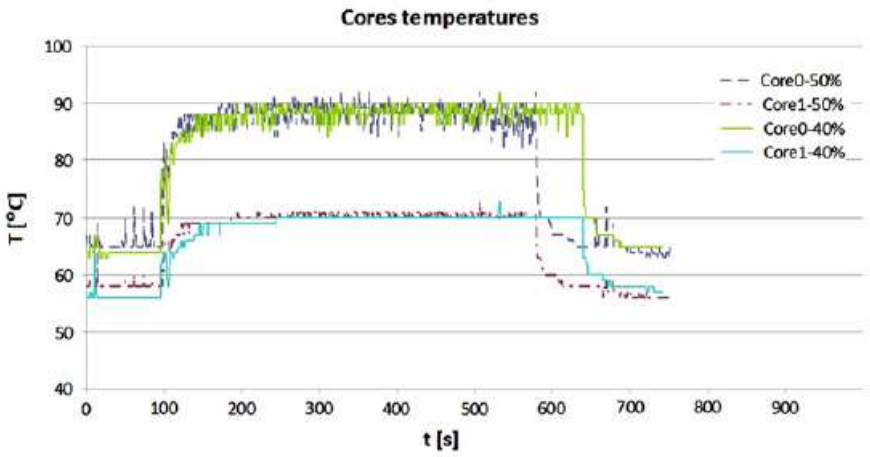

Fig. 24 Full performance and short sleeps workload implementations

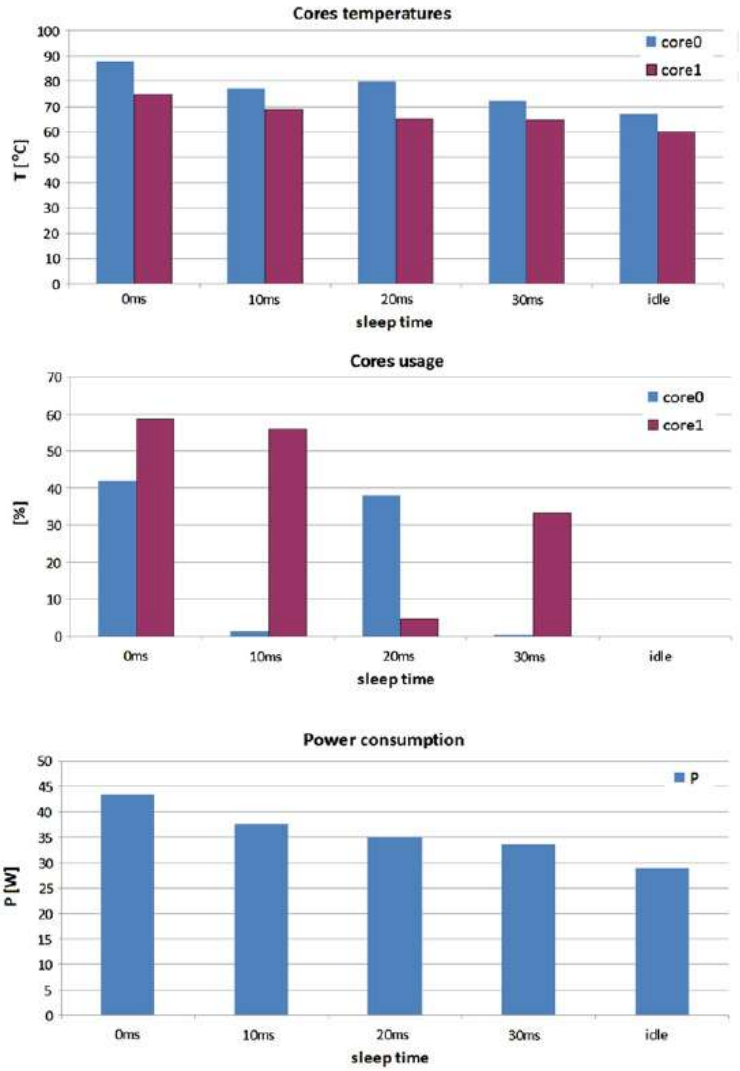

Fig. 25 Long sleeps workload implementations

By running the same test with different delay parameters, so that a workload is executed with different CPU usage levels, the plots in Fig. 25 have been obtained.

In Fig. 26(a) the same integer workload was run on core 0 at 100%, 90% and 70% CPU core usage levels. Four groups of dots are clearly observed: one cluster for the idle-state temperatures and three clusters for the workload temperatures. The same pattern can be observed for the relation between CPU usage levels and average CPU temperatures. There is a direct relation between CPU core temperatures and CPU time (in terms of usage level) for the same type of workload; therefore we can estimate the contribution of the application to the CPU heat generation.

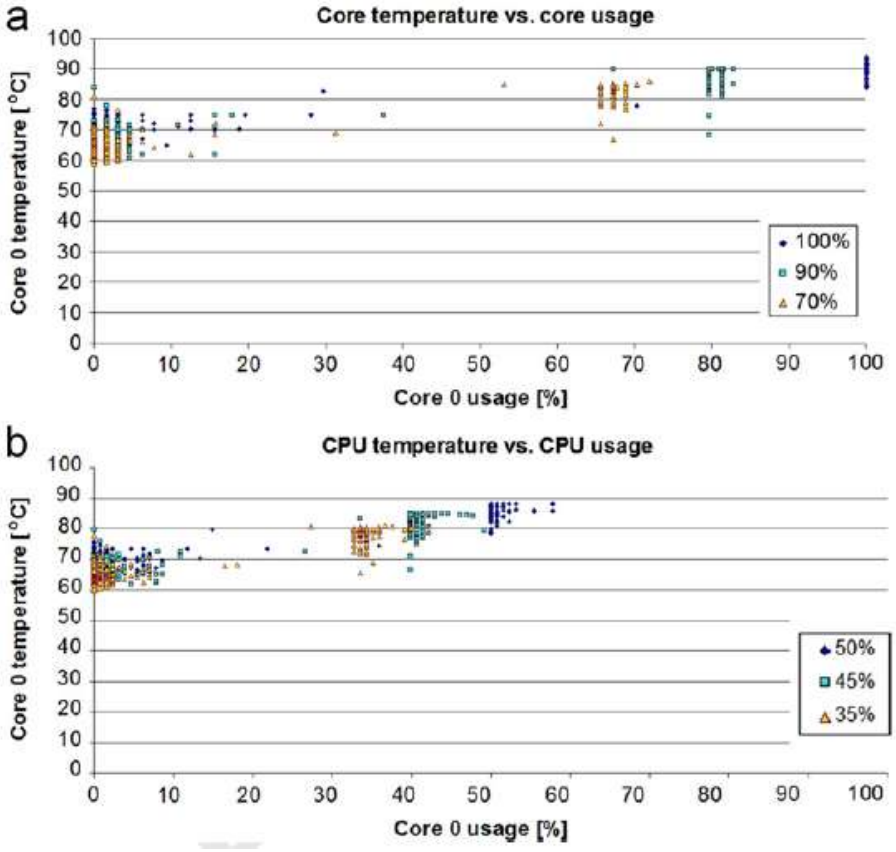

Fig. 26 (a) CPU core temperature versus CPU core usage and

(b) CPU average temperature versus CPU usage

Another aspect we investigated was the influence of different workload types on CPU cores temperatures. Fig. 27 depicts the thermal profiles of four workload types: integer, float, memory and SSE executed successively for the same amount of time on the same CPU core. Based on this test, in order to estimate the effect of a CPU intensive application over temperature, we need to know the type of operations the application implements.

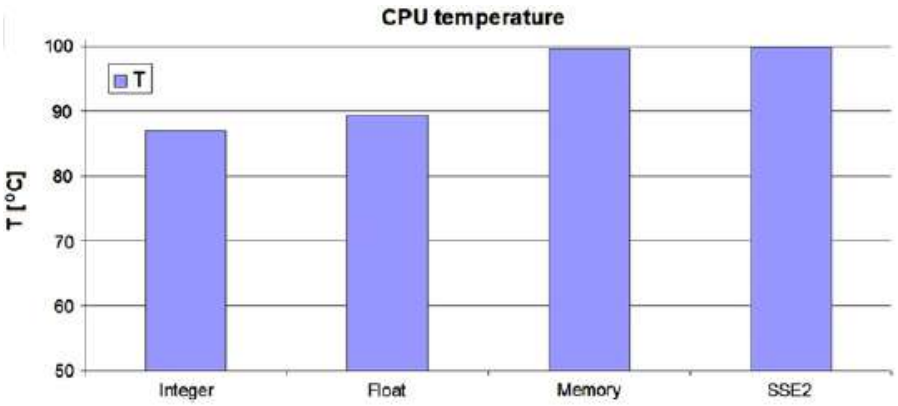

Fig. 27 CPU cores workload temperatures

Part of the battery energy of a mobile device is transformed into heat. The increase in temperature enforces more energy to be consumed. During the second phase of the benchmark application, when the workload is executed, the temperature of the processor and also the temperature of the entire mobile device increase (in our example the temperature increases from 60 to 100 ºC). This increase in temperature has an effect on power consumption, and a smooth increase (from 30 to 34 W) during phase 2 of the benchmark can be observed in the current profile. This increase of approximately 4W during the workload execution is due to heating of the device and available cooling elements.

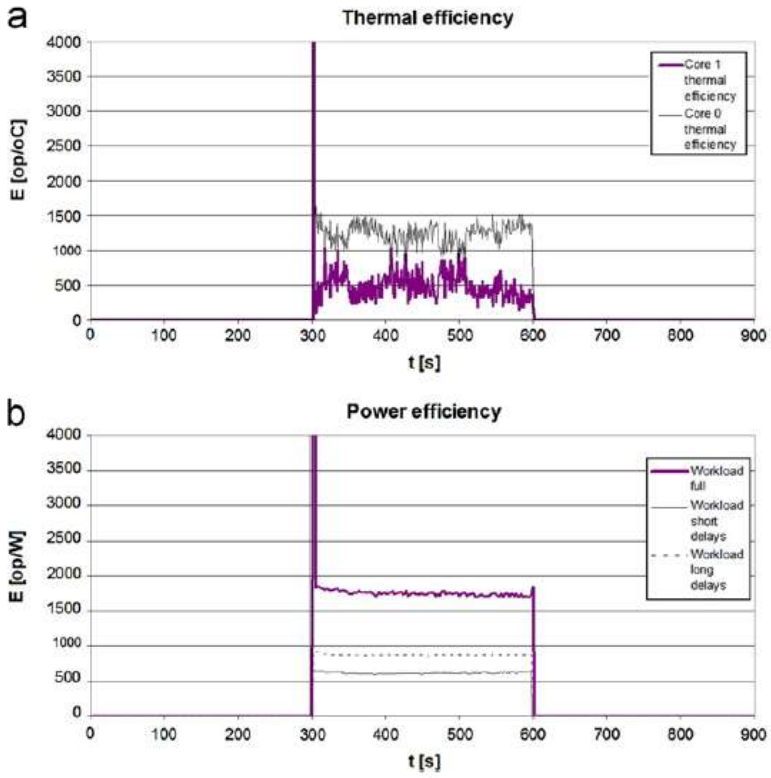

Fig. 28 (a) CPU cores thermal efficiency and (b) CPU workload thermal efficiency

We introduce and draw the thermal efficiency profile Fig. 28 as the number of workload (application specific) operations per increase in temperature. Thermal efficiency specifies how efficient is the use for every increase in temperature due to the execution of the workload in the selected parameters.

Based on the thermal profile and power consumption values of the mobile system achieved during tests we developed a power and thermal-aware application which adapts its execution in order to decrease CPU cores temperatures and increase the system's energy efficiency. In addition we investigated the possibility to identify the effect of every running application in the system over the CPU, cores and system temperatures.

2.5.4 Energy-aware video player

Nowadays the mobile phone is becoming the device people rely on not only for basic communication, but also for entertainment and even for work. Mobile devices have to run multimedia and communication applications together with user personal data and business applications that need optimum of energy-efficiency in order to prolog the battery lifetime. Therefore the last test application we implemented based on the energy framework is a video rendering system for a mobile device, which was tested on Pocket PC LOOX T380 Fujitsu Siemens. The proposed application adapts itself dynamically to the energy consumption of the mobile device modifying some parameters, which change some characteristics of the video rendering, to ensure a power usage reduction.

The power adaptation algorithm has been calibrated based on the following profiling results. Four benchmark movies have been profiled under several static and dynamic conditions, such as video encoding, file formats, locality or type of device. The first test targeted power consumption of video decoding algorithms. We played locally the same video and audio content encoded with different algorithms and file formats. The Cave video benchmark was played locally and the obtained power consumption values are shown in Fig. 29.

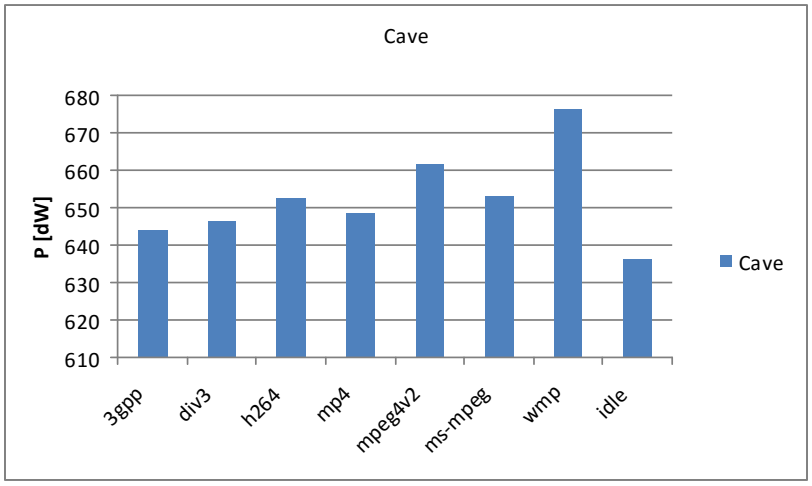

Fig. 29 Power consumption of the video decoder

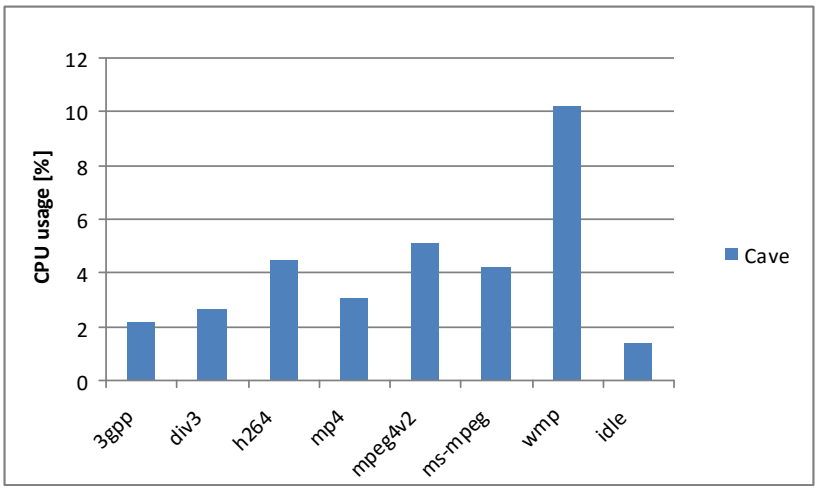

Fig. 30 CPU usage of the video decoder

It can be observed that Windows Media Video format and MPEG4 encoding consume more energy than the other codecs. We can also see that there is a direct relation between CPU usage level and its power consumption for every video format decoded (Fig. 30).

The second test evaluated the power consumption versus scene complexity or type of the video content. In this test we played locally every video benchmark encoded with the same codec and file formats. The Cave, Site, Tropics and Living video benchmark power consumption values are shown in Fig. 31. We used the four selected benchmarks encoded with MPEG-4 (AVI) and WMV 8 (WMV).

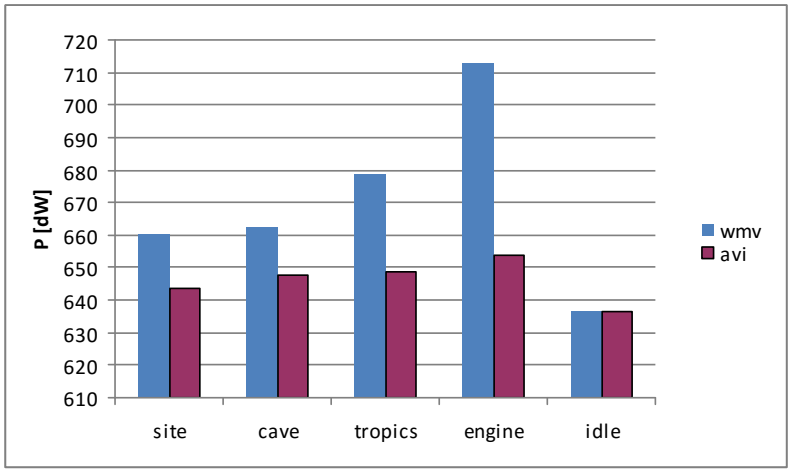

Fig. 31 Power consumption vs. video scene complexity

The impact of video locality (local or streaming playing) is presented next. In Fig. 32 local and streaming playing results is shown. It can be observed that when transferring the video benchmarks using WLAN, the overall power consumption increases but also the differences between power values are reduced because in that case is the wireless card.

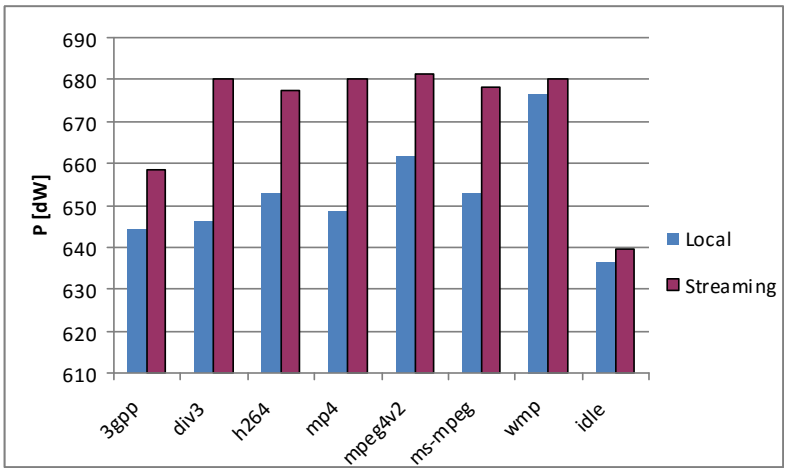

Fig. 32 Power consumption (local vs. remote)

Test cases executed on the mobile device gave the results presented in Fig. 33 for power consumption. It can be observed that multimedia video playing increase the overall power consumption four times the idle state power consumption.

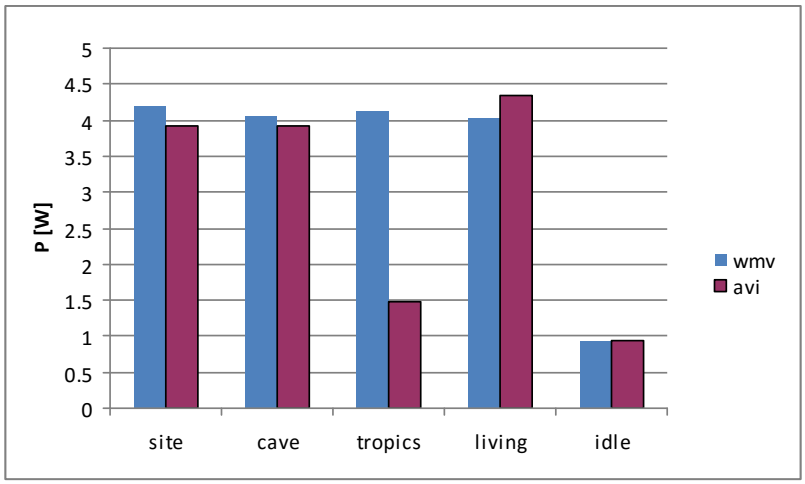

Fig. 33 Power consumption of mobile device

The power-aware player application was created in three stages: in the first stage the video rendering system was created, in the second stage the influence of some parameters of the system on the device's battery power level was studied and finally in the last stage the obtained results in the energy-aware video player application were applied. The creation of the system can be divided in two parts: the implementation of the video rendering part of the system and implementing the reading of the energy consumption of the application from the framework. The energy adaptation algorithm obtained during the evaluation and calibration tests is further presented. The parameters of the system that were taken into account were: the frame rate, the dimensions of the rendering window and the rendering frequency.

```
WHILE not exit condition
      READ battery status from framework
      IF (battery status <= 60%) AND (battery status > 50%)
            SET windows size to 0.66 of its original size
      ELSE IF (battery status <= 50%) AND (battery status > 40%)
            SET frame drop rate to 1/3
      ELSE IF (battery status <= 40%)
            SET frame display rate to 3
      END IF
END WHILE
```

When the device's battery level reaches the level of 60% from the total energy available when the battery is fully charged the parameter rendering window will be applied and the rendering window will be reduced by 1.5 from its initial dimensions. When the device's battery level reaches 50% the rendering window will be changed back to the initial dimensions and the frame rate will be multiplied by 3. After the device's battery level reaches 40% the parameter that ensures the best energy reduction will be applied, that is the rendering frequency. The frame rate will be established at the initial frame rate of the movie and the rendering frequency will be 3, indicating that only the frames from 3 to 3 seconds will be rendered. The parameter that ensures the best energy reduction is used last because when the battery's energy reaches a low level a better energy reduction is needed to ensure that the device will function a longer time.

Two video files were used to test the application and the proposed algorithm: one video file with the rendering time of 1 hour and the second video file with the rendering time of 2 hours. For the frame rate parameter there were made 12 tests: 6 tests for the frame rate multiplied by 2 and 6 for the frame rate multiplied by 3 (Fig. 34). For the rendering window's dimensions parameter there were made 12 tests: 6 tests when the window's dimensions are reduced by 1.2 in regard to the initial dimensions of the window and 6 tests when the window's dimensions are reduced by 1.5 (Fig. 35). For the rendering frequency parameter there were made 12 tests: 6 tests for rendering the frames and dropping frames from two to two and 6 tests for rendering one frame and drop 2 frames (Fig. 36). In addition to these tests there were made 6 tests considering the initial values for the three parameters: the frame rate 1, the specific frame rate of the movie, the rendering window's dimensions were the initial dimensions established using a panel as a parent for the window and the rendering frequency 1, indicating that all frames of the movie are rendered.

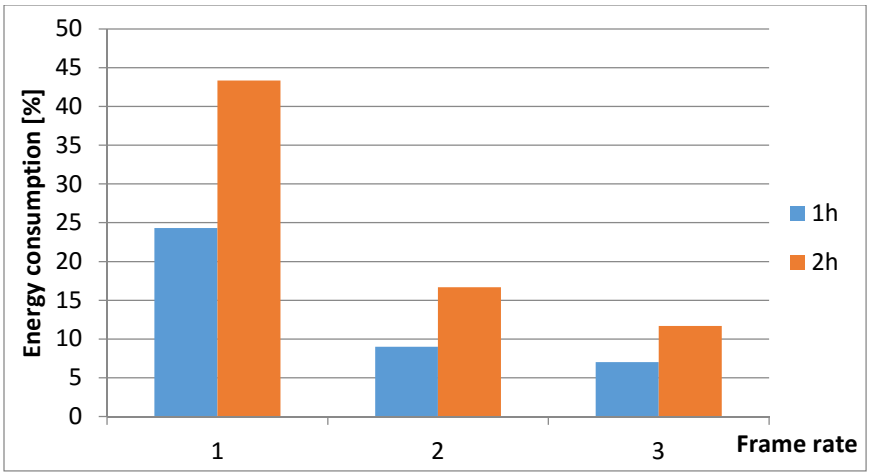

Fig. 34 Energy consumption vs. frame rate

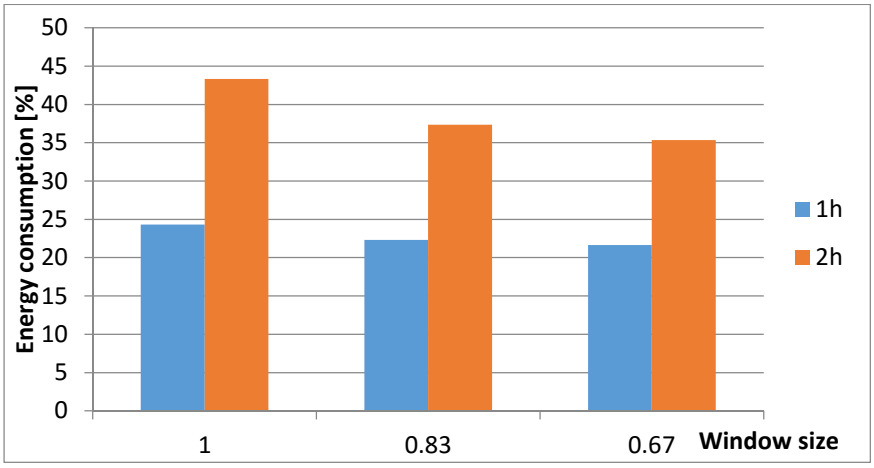

Fig. 35 Energy consumption vs. window size

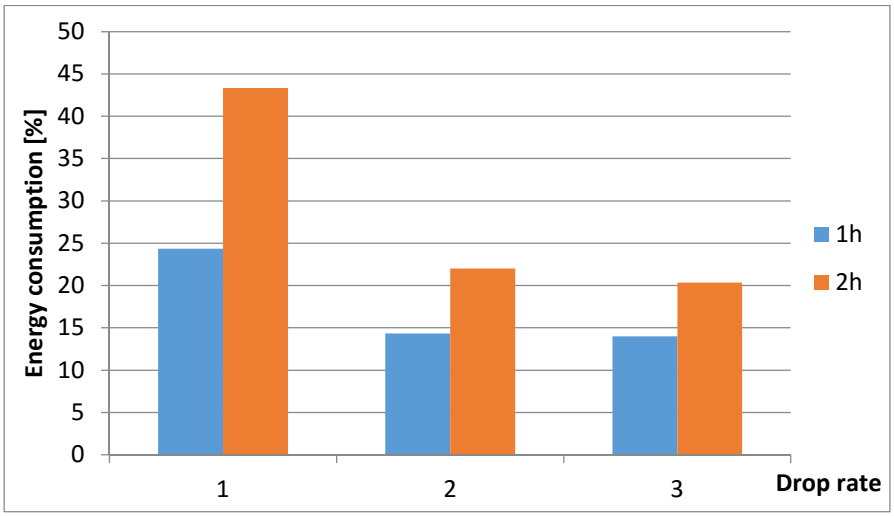

Fig. 36 Energy consumption vs. drop rate

2.6 RELATED WORK

The authors of [Chen2008] proposed and implemented a design framework called GreenRT, for developing power-aware soft real-time applications. They considered that the task in soft real-time applications must run quickly enough to meet the deadlines, but there is no extra benefit from running them faster than that so that the task finishes earlier. The energy could be saved if tasks are executed on a lower frequency in order to meet exactly their deadline, but not earlier than then their deadline. The GreenRT framework allows an application to monitor its progress and adjusts the processor frequency dynamically to finish its jobs in time. The framework is using the concept of DVFS implemented on modern processors which provides an energy efficient implementation for soft real-time tasks without any performance loss in respect to the user's perception [Chen2008]. How-ever the framework proposed by Chen et al. is not generic, it is highly dependent on the managed application.

During the last years, several similar solutions were proposed [Fei2008], [Cheung2009], [Bellasi2010] and [Vallina2011]. In [Fei2008] an energy-aware framework for dynamic software management of mobile systems has been proposed. The authors have designed a user space module, separated of the operating system, which permits the QoS (Quality of Services) based adaptation implemented at the application level. The energy consumption of the mobile applications is computed using predefined software macro-models which are hard to be implemented in practice for different types of applications. Unlike the solution adopted in [Fei2008] we estimate online the power consumption of the applications by measuring the battery and system's components parameters. The framework proposed in [Fei2008] was validated using three applications: video player, speech recognizer, and voice-over-IP. We obtained similar results for the video player energy-aware implementation using our framework.

In [Cheung2009] an energy optimization framework for software tasks is proposed. The work is focused on the optimization algorithm based on Markov decision process. The authors propose efficient techniques to solve the optimization problem based on dynamic programming and illustrate how it can be used in the context of realistic applications such as WiFi radio power optimization and email synchronization. Unlike the framework presented in [Cheung2009] our framework is oriented on monitoring and providing power consumption feedback to mobile applications. We have not compared our results with the ones achieved in [Cheung2009].

In [Bellasi2010] a cross-layer framework for power management of mobile devices is presented. The proposed framework involves different layers of a mobile system, from device drivers to user applications. The main goal of the framework is to aggregate the QoS levels of applications and select the optimum component specific power management policies according with the power requirements of the running applications. No similar application level experimental results were presented in [Bellasi2010] in order to compare the results.

In [Vallina2011] a user-centered energy-aware mobile operating system is presented. The proposed OS manages the hardware resources proactively allowing opportunistic access to external resources nearby. The work presented in [Vallina2011] follow another approach, using

an opportunistic access to computing resources available in nearby devices using local wireless interfaces and information about users' social networks.

In [Iyenggar2007] a new unified power management framework is proposed which describes a software architecture for a unified hardware and software controlled power management. Their framework provides a set of rules for classifying mobile device components in classes, each class having a set of well-defined states based on the running applications' states. Using these classes, the framework is an active software module that manages the power requirements of each application and appropriately handles the target components [Iyenggar2007]. Our work is similar with the solution proposed by Ivengarr et al. and have the same goals but is different because we try to use on-line power and system monitoring in order to provide a feedback for the control mechanism implemented in the framework.

In [Mittal2012] the authors present an energy emulation tool that can be used to estimate the energy usage of a mobile application. WattsOn extends existing emulators to estimate application battery consumption.

A more generic approach towards application-level prediction of power consumption and battery dissipation has been taken in [Krintz2004], where an estimation methodology is introduced. Based on a set of drain-rate curves from a set of benchmarks, estimation for an arbitrary program is composed. The estimation is limited to CPU and memory dimensions and it does not take into account the display, wireless communication or I/O components, but could be extended to include wireless basic benchmarks to assess wireless based communication applications.

VoIP usage is evolving for mobile devices such as smartphones with multiple wireless interfaces. However, the high energy consumption, presents a significant barrier to the widespread adoption of VoIP over Wi-Fi. To address this issue in [Agarwal2007] is presented a practical and deployable energy management architecture that leverages the cellular radio on a smartphone to implement wakeup for the high-energy consumption Wi-Fi radio.

In [Feeney2001], the authors investigate per packet energy consumption of an IEEE 802.11b card in various modes: transmit, receive, broadcast, and idle modes. Their data reveals how complexities introduced by the IEEE 802.11 protocol can impact the overall energy consumed by the card during its operation. The IEEE 802.11 WLAN specification describes a power-saving mode (PSM) that periodically turns the network interface off to save energy, and on to communicate [Krashinsky2005]. In the infrastructure mode a mobile device using PSM communicates with a wired access point (AP), the AP buffers data destined for the device. Once every beacon period, the AP sends a beacon containing a traffic indication map that indicates whether or not the mobile device has any data waiting for it. The mobile device wakes up to listen to beacons at a fixed frequency and polls the AP to receive any buffered data. Significant reduction in consumption is obtained using this static PSM method [Krashinsky2005].

2.7 SUPPORTING GRANTS, RESEARCH TEAM AND SCIENTIFIC OUTCOMES

Project title: Open Execution Framework for Low Consumption Battery Powered Mobile Devices

Principal investigator: PhD Marius MARCU

National competition: IDEI 2008

Implementation period: 2009-2011

Project acronym: MobiPower

Project code: 680/19.01.2009

Budget: ~100 000 EUR

Project summary: The evolution of portable and mobile computation systems towards an increase feature set and increasing hardware and software requirements demand, together with the significant increase of market penetration in our modern society, is raising complex problems from a reasonable energy consumption level point of view under different usage scenarios. Besides the need to assure a reduced energy consumption level from the user satisfaction point of view, the increasing number of mobile systems are contributing directly to the increase carbon dioxide emissions, which motivate the purpose of this project. In this project we aimed to define, implement and validate a software execution framework for mobile systems in order to reduce and optimize energy consumption. The role of the framework is to provide a generic interface for user applications such that applications can express and estimate their own energy consumption and to be able to adapt their execution in time in order to achieve an optimum energy consumption level. Applications that are going to use the execution framework in order to optimize their energy demands are called power-aware applications. The approach that we aim to pursue is both theoretical, from the energy or power consumption we aim to devise, and practical which reflects into the practical experiments we aim to conduct as well as the inter-disciplinary aspects it triggers. Some innovation elements of our project come from the theoretical model definition for the power consumption analysis, the model validation in multi-core, multimedia and wireless systems, as well through the process of applying our findings and theoretical concepts into the purpose to define a framework for power-aware application construction and promotion. At the end, our project aims to disseminate the results in order to promote and integrate its concepts in various productions products.

The research team involved in the project included experienced researchers, young researchers, external advisors and students:

Prof. Mircea POPA (scientific responsible)
PhD. Sebastian FUICU
PhD. Dacian TUDOR
Bogdan STRATULAT

Julia STRATULAT
Florin MATICU
Silvia COPIL
Camil MILOS

The main technical outcomes of the project are:

- Open execution framework for power-aware applications
- 3 power-aware applications:
  - Voice communication
  - CPU benchmarks
  - Video player
- Power-thermal profiling methodology
  - Power-thermal benchmarking
- Power-thermal profiling measurement database
  - CPU
  - WiFi IEEE 802.11 (a,g)
  - Multimedia

The main scientific outcomes of the project are (2009-2011):

- 1 patent request
- 45 papers: 3 ISI journal papers

This section is based on the following papers published by the author: [ISI2], [ISI4], [ISI9], [ISI11], [ISI18], [ISI19], [ISI23], [BDI13], [BDI21], [BDI24], [BDI28], [BDI32], BDI[38].

2.8 CONCLUSIONS AND FUTURE WORK

In this work we proposed a theoretical model for power consumption estimation of multiple components and multiple applications running on a single system. Based on this model we developed a portable software framework prototype for energy efficiency monitoring and dispatching between running user applications. We further run power profiling experiments for wireless interface, processor, and different types of applications and workloads. On top of the proposed framework and based on the obtained power profiles we developed four power aware applications: voice communication energy detection, processor intensive workloads energy detection, WLAN file transfer energy detection and video rendering application energy detection.

Based on the energy dispatcher theoretical model we developed the framework which is the main result of the project. The proposed framework is implemented on top of the host operating system and it provides energy monitoring and estimation services for those user applications that are adaptable and aware of the remaining battery lifetime. The main benefit of the power aware framework is to provide a platform for development of new kind of software applications which are adaptable to user requirements and available energy resources. The framework alone does not optimize the overall energy consumption of the mobile device but it provides the power-aware

applications with the means by which they can get an estimation of their own power consumption. Power-aware applications built on top of the framework are the final components responsible for system optimization decisions. In fact they dynamically find a tradeoff between quality of services provided to the users and energy required to execute these services. In order to validate the framework we developed several power-aware applications that shown an energy consumption optimization range between 5 to 60 %. These applications are further presented in the results section of the document.

The second important result of this research work is the power characterization of the main hardware components of a mobile device: processor, memory, wireless chipset, GSM chipset and display. Using the power characterization process of system's components the power profiles or signatures of software workloads or algorithms implementations were further extracted. The power framework implements the online profiler for power consumption of system's components and running applications.

3 ENERGY PROFILING OF VIRTUALIZATION SOLUTIONS

3.1 OVERALL DESCRIPTION AND RESEARCH OBJECTIVES

The environment sustainability is one of the main directions of intervention for all developed countries. Solutions for alternative energies generation and energy reduction are already discussed. Furthermore, energy efficiency has become an important aspect in data centers and large server systems. Virtualization is one of the main research directions for both large scale data centers and servers. This section explores how virtualization influences the power consumption of both physical systems and virtual systems and which is the most efficient way to implement such applications.

Power consumption is a critical design parameter in modern data-center and enterprise environments, since it directly impacts both the deployment (peak power delivery capacity) and operational costs (power supply, cooling). The energy consumption of the compute equipment is a major component of these costs. [Carpenter2007, Dhiman2009]

Modern data centers use virtualization (e.g. VMware), to get better fault and performance isolation, improved system manage-ability and reduced infrastructure cost through resource consolidation and live migration [Clark2005]. Consolidating multiple servers running in different virtual machines (VMs) on a single physical machine (PM) increases the overall utilization and efficiency of the equipment across the whole deployment. However, as we show later on, based on the utilization levels and characteristics of these different co-located VMs, the overall power consumption and performance of the VMs can vary a lot. This can create hot-spots of activity, and degrade overall performance and energy efficiency. [Dhiman2009, Tian2007]

A new trend in virtualization is to use it for mobile and embedded multi-core architectures [Behmann2009].

The main goal of this work has been the study on the power consumption impact of virtualization solutions for common desktop and laptop computers. This work explored how virtualization influences the power consumption of both physical systems and virtual systems and which is the most efficient way to implement such applications.

3.2 THEORETICAL CONCEPTS

We aim to measure power consumption and heat dissipation of several VMs hosted by one PM. Power consumption of the PM and the temperature of specific internal components of the PM (e.g. HDD, CPU cores), can be measured using existing built-in sensors and external meters. However, power consumption and temperature of a VM cannot be measured in hardware, but they can be estimated using virtual instrumentation and measurable VM parameters. Power consumption of a PM (noted P_{PHY}) is a hardware measurable parameter which quantifies the power consumption of the whole PM system while running. PM power consumption contains two components: static

power consumption of the PM system when idle and dynamic power consumption of the PM due to the software workloads executed on the PM, as in (PPHY = PIDLE + PDYN (Eq. 14). The dynamic component of P_{PHY} is the supplementary amount of power needed by the system's components to execute the specific workloads of the running $\boldsymbol{PDYN=iPAPi}$ (Eq. 15).

$$P_{PHY} = P_{IDLE} + P_{DYN} \quad \textit{(Eq. 14)}$$

$$\boldsymbol{P_{DYN} = \sum_i P_{APi}} \quad \textit{(Eq. 15)}$$

The dynamic power consumption of a PM hosting several VMs, described in ($\boldsymbol{PDYN=\sum_i P_{VMi}}$ (Eq. 16), contains the additional amount of power required to execute the workload tasks of the running VMs.

$$\boldsymbol{P_{DYN} = \sum_i P_{VMi}} \quad \textit{(Eq. 16)}$$

Power consumption of a VM (noted P_{VMi}) is a virtual parameter that quantifies the additional power consumption of the PM due to the tasks executed by the VM. P_{VMi} can be estimated based on power models for VMs, or from a measurement scheme like the one we propose in this work. Temperature is another important parameter for both physical servers and virtualization solutions. The temperature can be measured for various PM internal components such as built-in sensors. We addressed the temperature of CPU cores (noted T_{COREi}) and the average CPU temperature (noted T_{CPU}) for PM. Virtual CPU temperature (T_{VCPUi}) is the temperature increase of the physical CPU due to the tasks executed by the VM_i.

Each test in a test sequence consists of three phases (Fig. 37). The test starts with an idle state waiting period which brings the system's parameters to idle static values. During the second phase the test workload is executed. After workload execution the second idle waiting phase is provided for system's parameters to reach back the initial static values. Every test in a test case last for the same amount of time. The time of phase 1 is constant for the tests along a test case and the duration of phases 2 and 3 together is also constant.

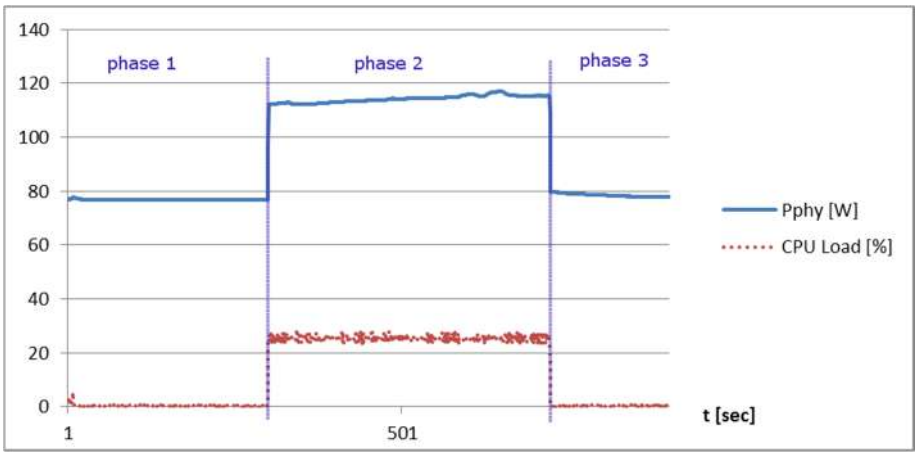

Fig. 37 Test phases

Several parameters were measured, computed and evaluated during tests execution. These parameters are described and defined next and they are shown in Fig. 38 and Fig. 39. Power consumption and energy parameters are described in the test representation shown in Fig. 38. P_{min} is the average power consumption of the PM when idle (P_{IDLE}), it is measured in Watts and it is measured in the first phase of every test. P_{avg} is the average power consumption of the PM when running the workload application. P_{avg} is measured and computed during phase 2 of a test execution. P_{dif} is the difference between maximum workload power consumption and the idle power consumption. It is computed at the end of phase 2 of each test. P_{thm} is the difference between P_{max} and the average of first power consumption samples of the workload phase of the test. P_{thm} is an indicator of the increase of power consumption of workload phase due to the temperature increase of the CPU. We have computed the energy consumed by the whole test execution using two components: E_{idle} and $E_{workload}$. E_{idle} is the energy consumed by the PM considering the whole test is idle. $E_{workload}$ is the additional energy consumed by the PM to execute the workload. The energy values are represented in Joules.

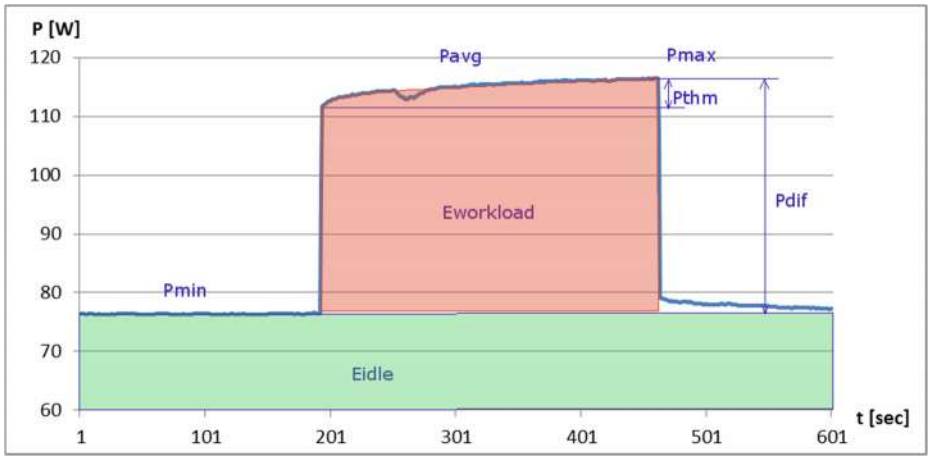

Fig. 38 Power and energy parameters

The same parameters described for temperature and heat energy dissipation are presented in Fig. 39. T_{min} represents the temperature of the system components (mainly the CPU) when the system is in the idle state, before any workload is executed. T_{max} is the maximum temperature measurements when executing the workload – usually depending on thermal benchmark calibration it should be the steady state temperature of the workload.

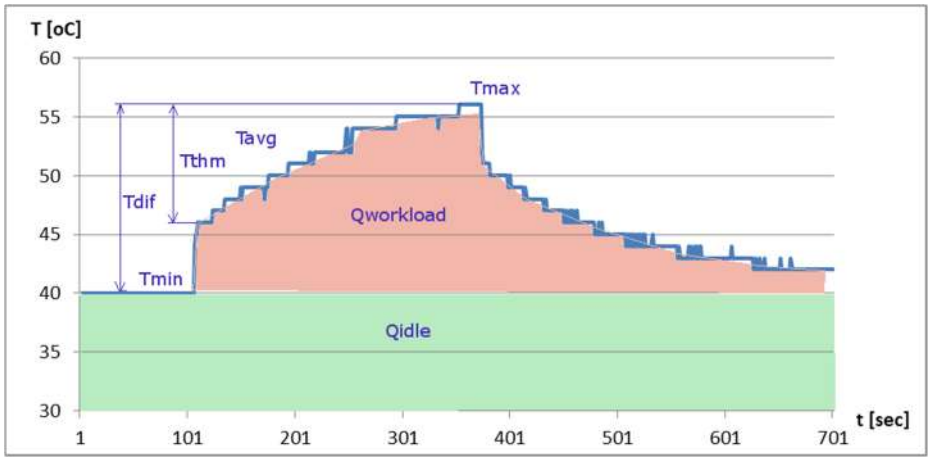

Fig. 39 Temperature and heat parameters

3.3 MEASUREMENT TEST BENCH

We want to estimate the consumption and temperature of a virtual machine using a measurable test workbench. The overall architecture of the proposed workbench is shown in Fig. 40. It contains the PM under test and VMware virtualization solution was used to install and deploy four VMs on both PM operating systems. All VMs under tests have Debian Linux 5.0 operating system installed and every VM owns one physical CPU core and 1 GB of memory (Fig. 41). An external computer was used to monitor power consumption of PM under test and to collect and store measurement values from PM and VMs. Power consumption of the PM was measured using an AC power meter, that continuously send available measurements to the monitoring station using wireless interface with a sample rate of one per second. An external temperature sensor is connected to the monitoring computer serial port and is used to monitor environment temperature. Systems under tests we selected were one laptop and one desktop.

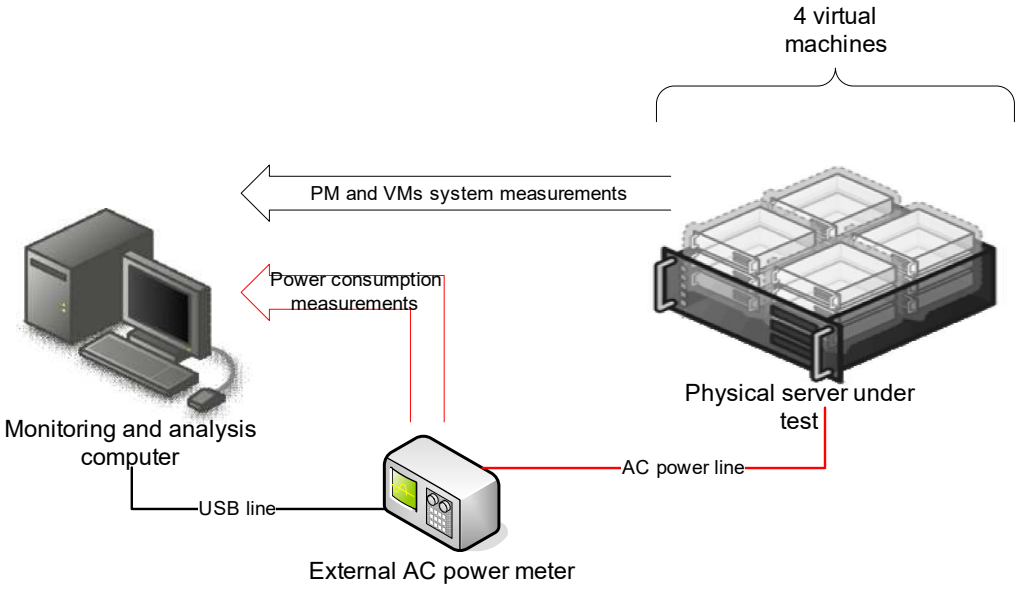

Fig. 40 Overall measurement architecture

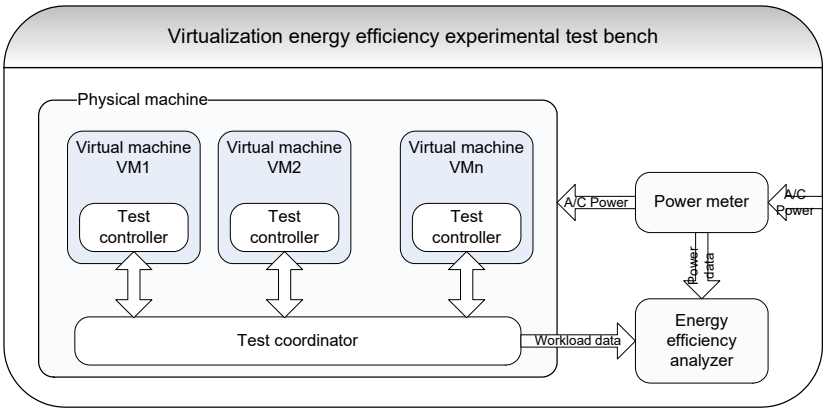

Fig. 41 Overall evaluation setup architecture

In order to emphasize the power consumption due to virtual machine execution and to estimate energy efficiency of the virtualized tasks, several test cases were defined. Every test case contains a sequence of benchmark applications with different combinations of parameters called test workloads (Fig. 42). Running a test case involves the execution of each test workload from that test case three times on each of the following machine combinations: Windows PM, VMs hosted by Windows PM, Linux PM and VMs hosted by Linux PM. In the Fig. 42 two parameters, power consumption and CPU load, measured during test execution, are presented. It can be easily observed the parameters variation during workload execution of every test.

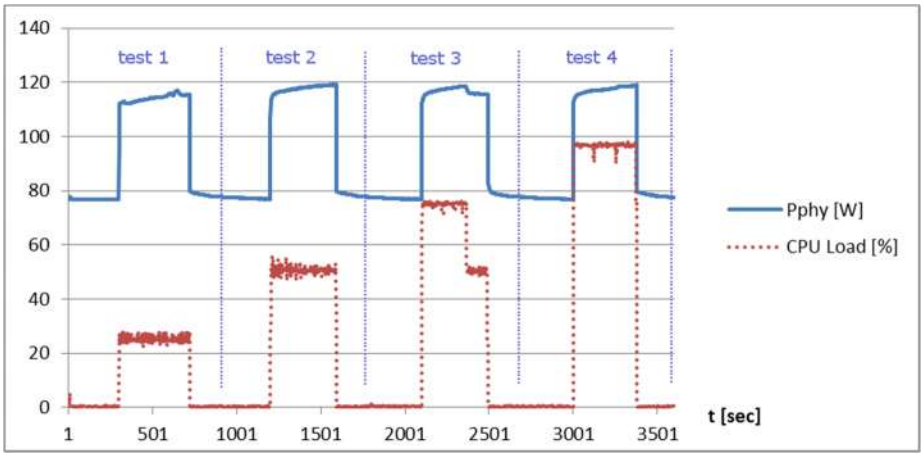

Fig. 42 Test sequence in a test case

3.4 EXPERIMENTAL RESULTS

3.4.1 Idle state physical system and virtual machines power consumption

The first test case we run on every system under test was introduced to estimate power consumption of the physical system when running in idle state on the host operating system. We

consider that the system idle power consumption is important in because the workloads' power consumption introduced in the next test cases will be estimated compared with this initial value. Furthermore, the test is needed to investigate the accuracy and repeatability of the measurements.

The second test case is specified to estimate power consumption of the physical system when running one or more VMs in idle state under different configuration parameters. During first two tests no workload application was executed. The conditions specified for this test are: number of VMs started and their parameter settings, the number of CPU cores and the size of RAM allowed for one VM.

In the beginning, the idle state power consumption and temperature measurements for VM and PM were performed and analyzed according with first two test cases (Fig. 43). It can be observed that when PM is idle, the Windows OS consumes less power than the Linux OS. In fact this depends on the number of services running of both measured systems. In the proposed tests two clean installed OS were used with default services. The difference is approximately 10% in power consumption levels between Windows and Linux OS. An increase of power consumption of the system is noted when starting the VMs (4 VMs) and all of them are idle. Under Linux OS the observed increase of power consumption when running the VMs is rather small. The difference is approximately 1% (~1W). But when the system is running Windows OS the increase of power consumption when VMs are on is significant higher. The difference is approximately 9% (~7W). We run the tests on two different machines a desktop and a laptop computer (Fig. 44).

	Power consumption [W]		**Temperature [^{o}C]**	
	Linux	**Windows**	**Linux**	**Windows**
PM	77.05	70.01	39.91	40.67
VM	77.93	77.05	41.31	44.2
Increase [%]	1.13	9.14	3.39	7.99

Tab. 2 Idle power consumption and temperature values

An increase of average CPU temperature is observed when running VMs compared with the PM temperature. The higher increase in temperature is observed on Windows, approximately 8% (3.5 ^{o}C) as shown in Tab. 2.

Based on these tests we considered that is important to propose and select an appropriate test execution strategy in order to overcome the differences in power consumption between both operating systems and hardware systems. Absolute power consumption values cannot be considered in our test, therefore every test follows the same pattern in order to emphasis the relative power values between idle states and workload states. Energy values should be compared considering the system performance and number of workload operations, thus we computed the energy efficiency of a test workload execution.

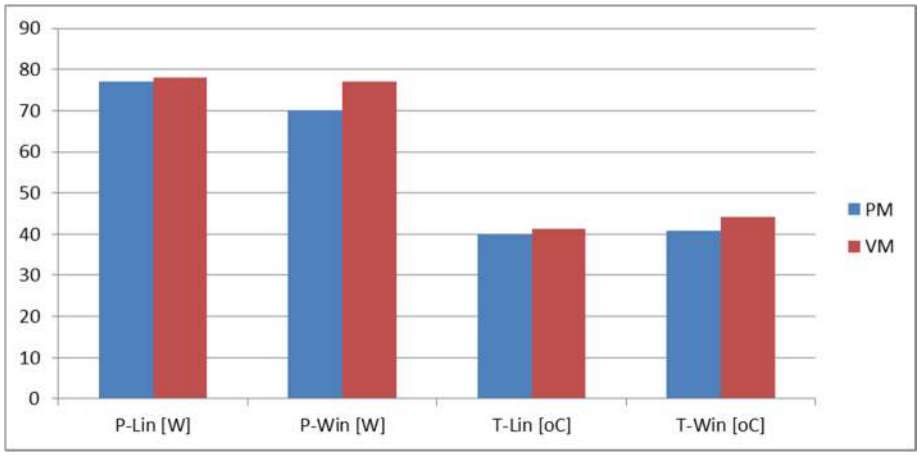

Fig. 43 Idle PM and idle VM power consumption and temperature

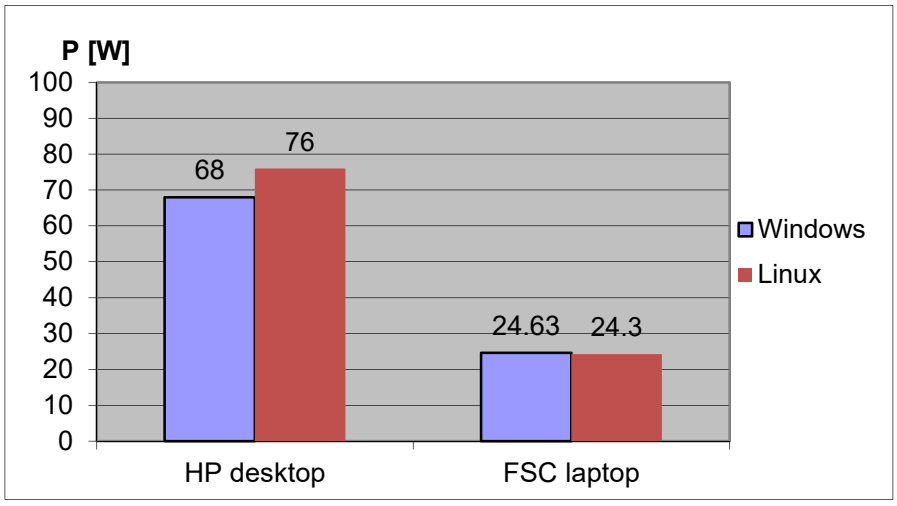

Fig. 44 Power consumption of physical systems under tests

3.4.2 CPU and memory workload virtual machines power consumption

The third set of test cases is performed in order to estimate power consumption of the physical system when running one or more VMs and each VM executes certain workload. This test case tries to estimate how physical system's power consumption varies with different types of CPU workloads or benchmarks when running on one or more VMs compared to running directly on the physical system. For the workload phase of the test sequence we used different CPU and memory benchmarks: integer, memory and floating point. Every test execution was parameterized with the following settings: the number of VM instances, the running VM settings (CPU cores and memory size), and the number of simultaneous workload instances (processes or threads). Every test case was executed once on the physical system and then on the selected VMs. For every workload benchmark we logged also its performance data in order to estimate the energy efficiency for every test condition.

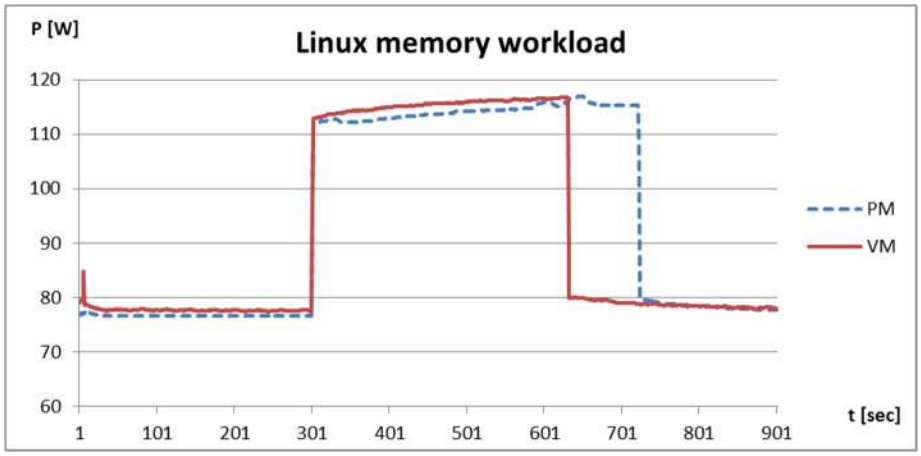

Fig. 45 Linux memory PM and VM workload power consumption

We have considered two more memory test cases. The first one compares the energy efficiency of the same task executed on different machines and operating systems: Windows PM, Windows VM, Linux PM and Linux VM. Power consumption profiles of the test workload executed on Linux are shown in Fig. 45. It can be easily observed that the VM execution is more energy efficient than the PM execution and it is also best performing. This is a strange observation that we cannot explain and it is repeated for all the tests with single task workloads. Power profiles obtained after test execution under Windows OS are shown in Fig. 46. Task execution on the VM in this case is slower than it execution directly on the PM.

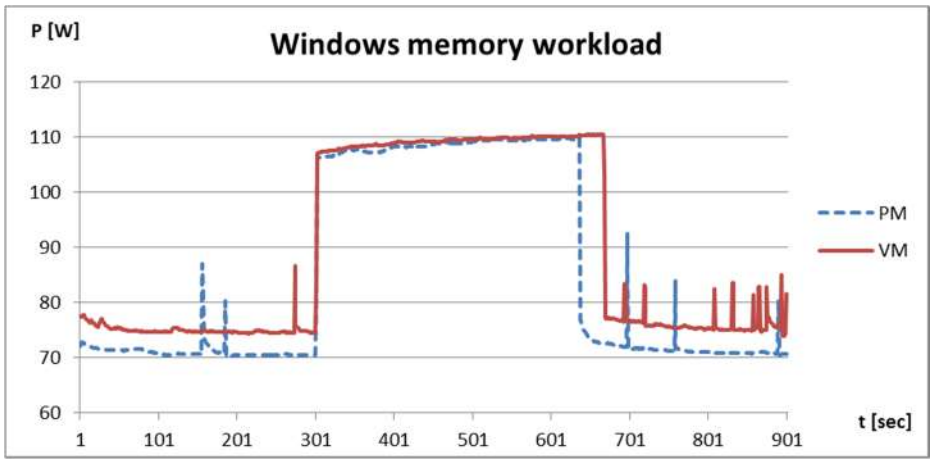

Fig. 46 Windows memory PM and VM workload power consumption

Thermal profiles for this test case execution are shown in Fig. 47 for Linux OS and in Fig. 48 for Windows.

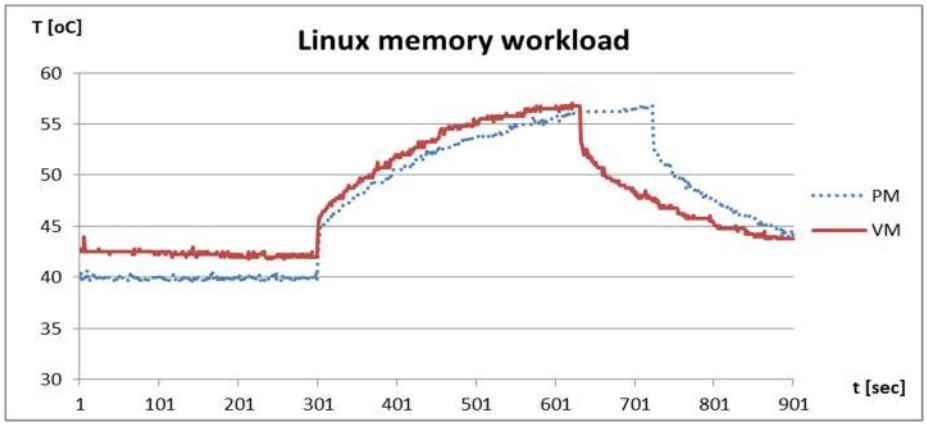

Fig. 47 Linux memory PM and VM workload average CPU temperature

In order to compare the four test execution in different environments, their energy efficiencies were computed. The energy efficiency of the RAM transfers workload is calculated dividing the workload energy ($E_{workload}$) to the side to data. Energy efficiency is measured as number of bytes per unit of energy consumed (Fig. 49). Thermal efficiency of the VM/PM test execution is shown in Fig. 50.

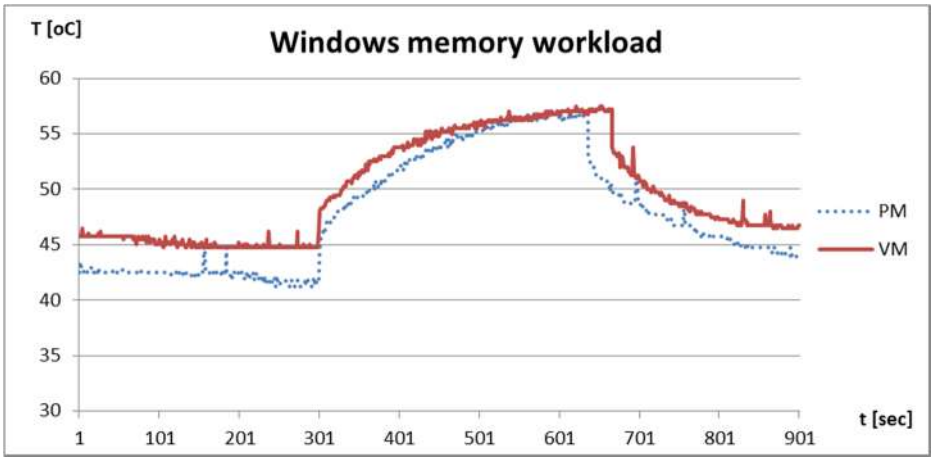

Fig. 48 Windows memory PM and VM workload average CPU temperature

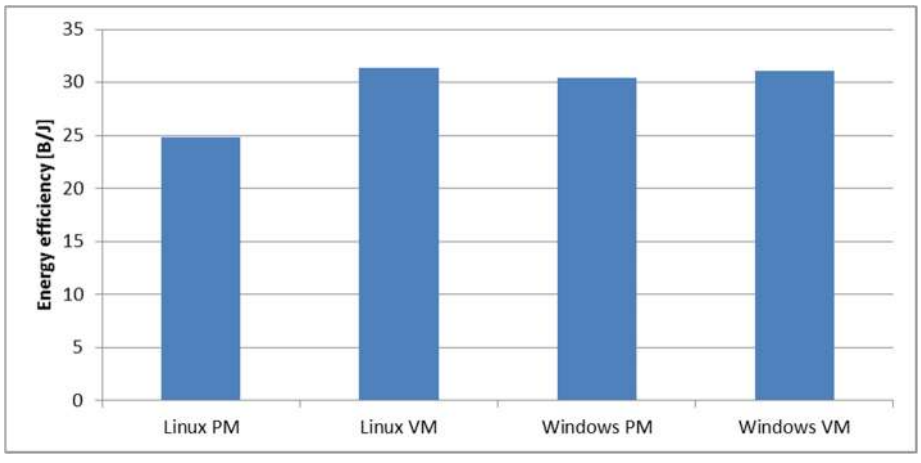

Fig. 49 Energy efficiency of memory transfers

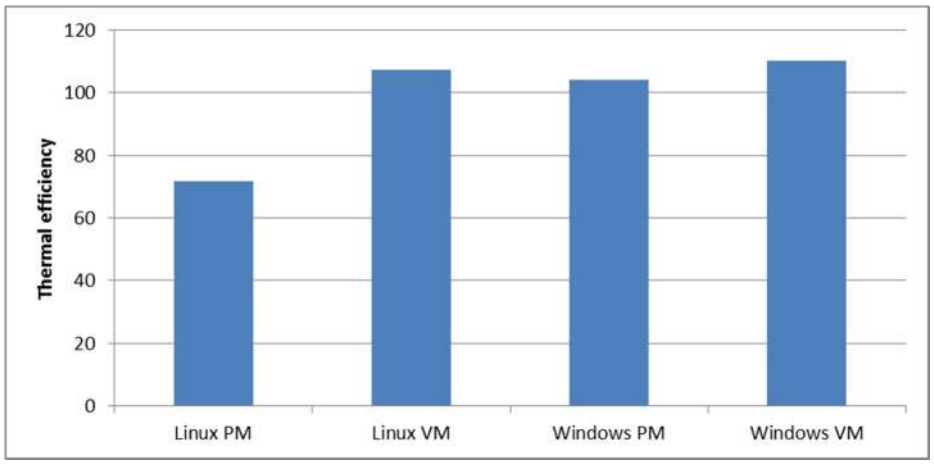

Fig. 50 Thermal efficiency of memory transfers

The second test emphasizes the variation of PM/VM parameters with various number of test instances running simultaneously. The average power consumption values of test workloads when executed multiple test instances are shown in Fig. 51. The same workload is executed as one instance which means one single task running on PM or VM. Having two test instances means that the whole workload is split in two equivalent applications that are executed simultaneously as two processes on PM or two distinct VMs.

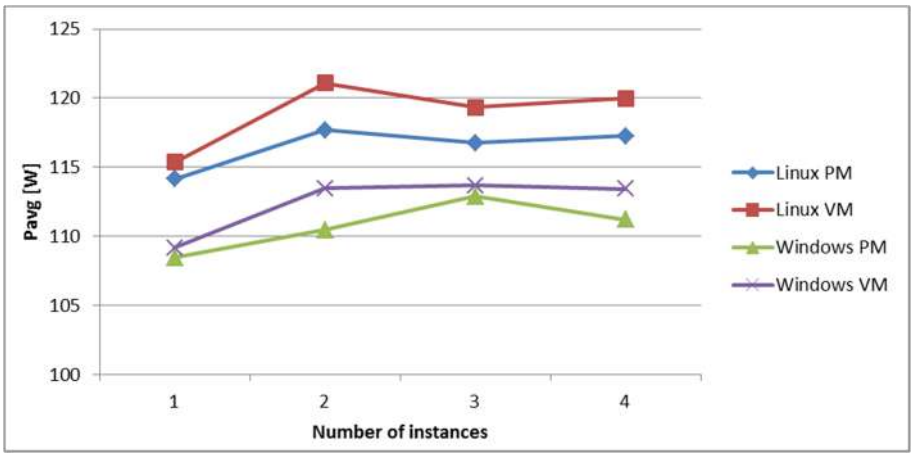

Fig. 51 Average power consumption of memory workload

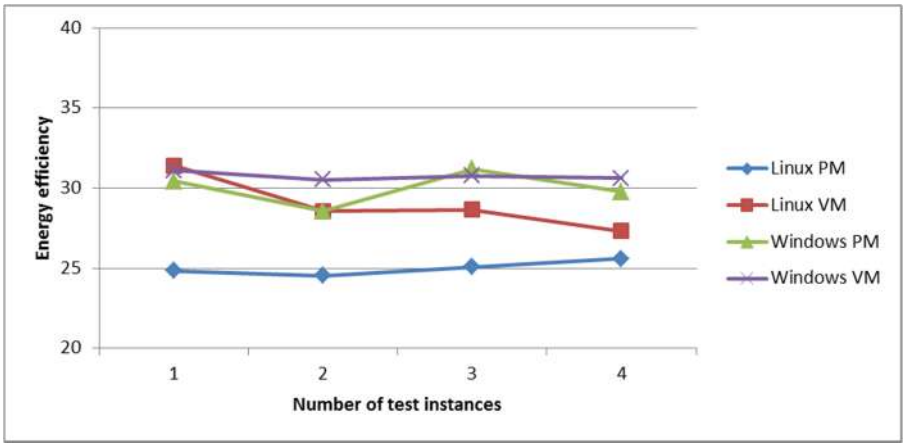

Fig. 52 Energy efficiency of memory workload multiple instances

The energy efficiency of the multiple instances test executions is shown in Fig. 52. Constant or small decreasing of energy efficiency is observed when running tests on the increasing number of VMs

3.4.3 I/O workload virtual machines power consumption

The fourth set of test cases is specified to estimate power consumption of the physical system when it executes one or more I/O workloads within VMs. With this test case we try to show how virtualization influences the I/O operations. One important aspect we want to cover with this test case is hard disk I/O workload using existing disk benchmarks. Other I/O test cases could also be implemented like USB, video, sound, etc. In our test we ran only DISKIO test cases using a disk benchmark, parameterized with the following settings: the number of VM instances, the running VM settings (CPU cores and memory size), and the number of simultaneous workload instances (processes or threads).

The two test cases within this test set were selected to characterize power consumption and temperature of I/O intensive workload. Using these tests the PM and VM disk I/O workload can be compared. Power consumption of I/O operations have been performed using hard disk benchmarking, using a file system benchmark tool running on different platforms that generates and measures a variety of file operations: read, write, re-read, re-write, read backwards, read strided, fread, fwrite, random read. We executed the I/O workload with different parameters on both operating systems under tests and both types of machines (PM and VM). Power measurements for one test execution are shown in Fig. 53 and temperature profiles are presented in Fig. 54.

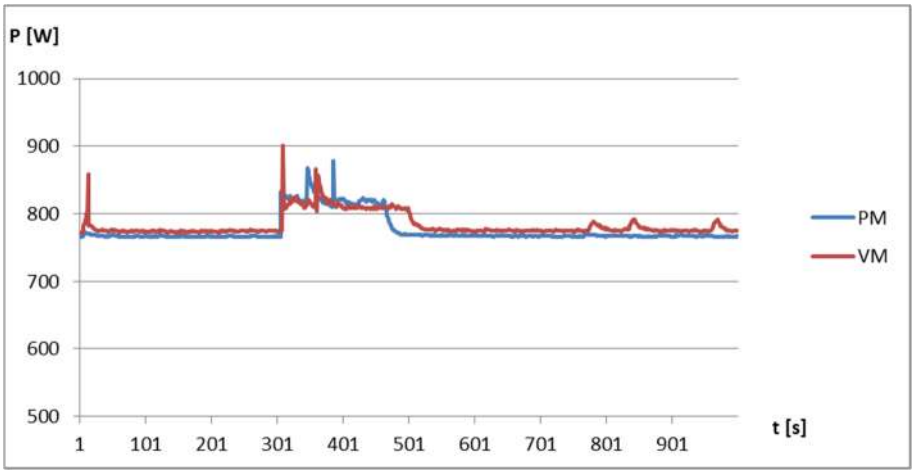

Fig. 53 Power profiles of I/O workload tests

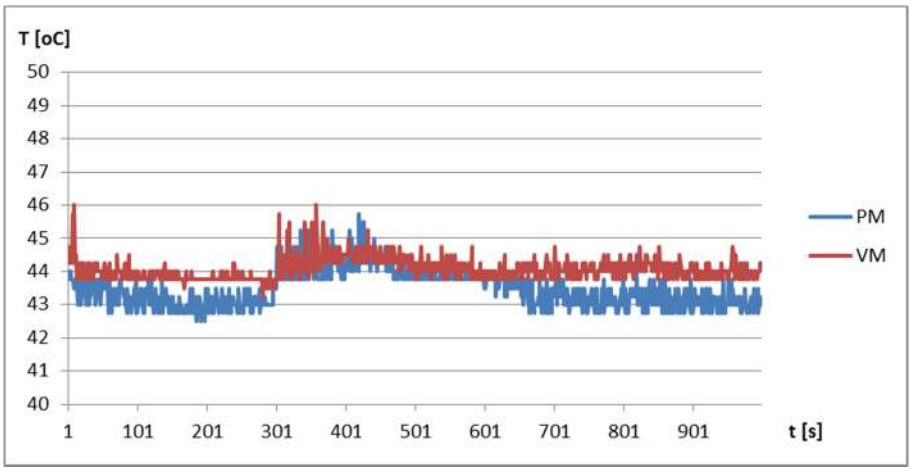

Fig. 54 Temperature profiles of I/O workload tests

10 % increase in energy consumption for the virtualized task execution compared with the PM one, is observed during the tests. However, the temperature increase during the I/O tests is moderate (2-3 ^{o}C), which is similar for PM and for VM.

In Fig. 55 the concluding results of energy efficiency of CPU, memory and disk I/O tests are shown. Due to the implementation particularities of L4Linux memory operations are less preformat than non-virtualization implementation therefore the energy efficiency is lower with ~25%. The disk I/O operations however are similar on both platforms Ubuntu and L4Linux. Another observation is that L4Linux implementation consumes less power than Ubuntu, when idle, on the same machine.

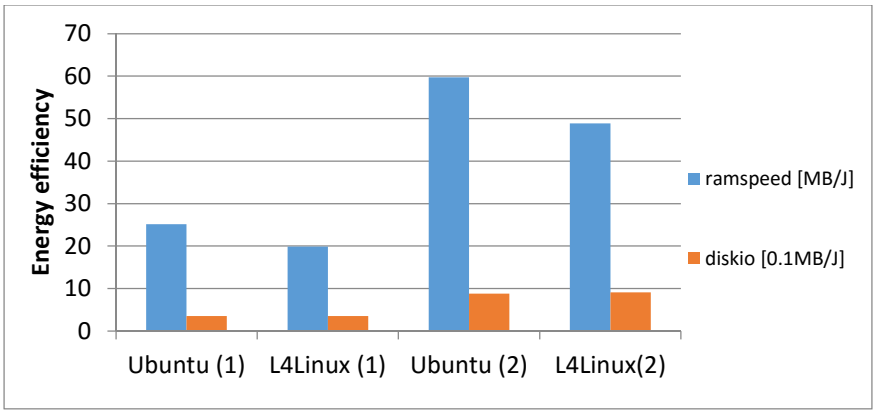

Fig. 55 Memory and I/O operations energy efficiency

3.5 RELATED WORK

A cornerstone in the energetically evaluation for virtualized systems is the measurement procedure and context for both physical systems and virtual machines. Power consumption of physical servers is an important metric used when evaluating different virtualized solutions implemented on top of these servers. The power consumption issue of computing systems is in general a very complex one because every physical component in the physical system has its own power consumption profile depending especially on its execution workload. In virtualized environments the power consumption modeling problem is much more complex because software applications are running on VMs and they do not access directly the physical components. The host operating system has to provide access to physical components and share these components for different VMs and their applications. The nature of workload executed in each VM determines the power profile and performance of the VM, and hence its energy consumption [Dhiman2009]. The complexity of measuring energy efficiency for virtualized systems is increasing with the number of elements that should be addressed (e.g. number of VMs, OS, PM, power management mechanisms activated, software applications running on VMs).

The author of [Carpenter2007] proposed and performed a set of virtualization performance tests for three types of Intel multi-core based servers in order to estimate whether their virtualization can deliver significant benefits in data centers over non-virtualized servers. During the performance tests overall power consumption was measured and power consumption per workload was computed in order to determine the costs of providing the requested level of performance. Virtualization enables one to consolidate multiple workloads onto each server, increasing utilization and reducing power consumption per workload [Carpenter2007]. A CPU intensive complex database application was used as testing workload, and they progressively increased the number of virtualized workloads. In our approach we use three types of simple operations as workload in order to address the main components of the system: CPU, memory and I/O.

Another important element in energy efficiency evaluation for virtualized systems is related to the power management mechanisms and their implementations. The authors of [Dhiman2009] presented a multi-tier software solution for energy efficiency computing in virtualized

environments based on the characteristics of the workloads co-located on the same PM. The paper shows that co-location of VMs with heterogeneous characteristics on same PM is beneficial for overall performance and energy efficiency. In [Verma2008] the authors investigate the design, implementation, and evaluation of a power-aware application placement controller in the context of an environment with heterogeneous virtualized server clusters. Their solution dispatches applications to different VM or PM taking in account performance requirements, migration costs and power consumption. The tests and experiments were executed based on the traces obtained from server farm of a large data center.

In [Tian2007] specific work related to power management of virtualized OS is presented. The authors tried to map virtual ACPI power states of VM components (e.g. CPU P-states, OS S-states and devices D-states) to real power states of the PM in order to increase the efficiency of overall power management mechanism. Nathuji and Schwan explored in their work how to integrate power management mechanisms between VMs and host PM while keeping isolation between them [Nathuji2007]. They proposed and implemented a software solution called VirtualPower which extends the VM power states and assign specific power policies to these states. Their main challenge is again to map VM power states to real power states of the PM.

The authors of [Ye2010] focused their research work to power management of I/O disk operations in virtualized environments. This paper presents three proposed improvements to address the disk's device drivers' power states mapping between VM and PM, based on the statistics of I/O activities between PM and VM. Their solutions are based on different combinations between buffering mechanism in the PM that buffers writes from the VMs and early flush mechanism that flushes the dirty pages from the guest OS buffer caches prior to putting the disk to sleep.

A major challenge in computer systems is the coexistence of real-time and non-real-time applications on the same machine. The authors of [9] describe the microkernel architecture of L4 and which provides both virtualization and real-time support. On a real-time capable microkernel, all applications are temporally isolated and can execute with real-time guarantees even they are virtualized. L4 Linux is a para-virtualization solution which requires changes in the guest operating systems in order to run in user space of the CPU. The changes are required in platform-specific code of Linux but all other code and device drivers are unchanged. L4 Linux was ported on IA-32 and ARM processors architectures; therefore it may be used in the near future on next multi-core mobile devices.

3.6 SUPPORTING GRANTS, RESEARCH TEAM AND SCIENTIFIC OUTCOMES

Project title: Embedded Multi-Core Processing for Mobile Communications

National competition: FP7-ICT

Implementation period: 2008-2010

Project acronym: eMuCo

Project code: 216378/01.02.2008

Budget: 147 000 EUR

Project summary: The aim of the ICT-eMuCo project is to develop a platform for future mobile devices based on multi-core architecture. This comprises the relevant controller element as well as the operating system and application layers. A multi-core architecture has been used to obtain the best ratio of performance and power consumption while maintaining the flexibility and scalability of the system through variations in number of cores, cache sizes, or clock speed. Virtualization technologies have been used to abstract the applications from the hardware layer architecture. The project has generated a series of results which can be seen from three different perspectives: software platform perspective, hardware platform perspective, and application layer perspective.

The research team involved in the project included experienced researchers, young researchers, and external advisors:

Prof. Vladimir CRETU (local partner director)
PhD. Dacian TUDOR
PhD. Georgiana MACARIU
PhD. Calin JEBELEAN
PhD. Marius MARCU
Prof. Mihai MICEA

The main contribution to the project is the study on energy and thermal efficiency of virtualization solutions implemented on the two OS used today: Windows and Linux. In order to achieve this result the evaluation methodology and measurement setup have been proposed and implemented.

This section is based on the following papers published by the author: [BDI14], [BDI16], [BDI17].

3.7 CONCLUSIONS AND FUTURE WORK

This section explores how virtualization influences the power consumption of both physical systems and virtual systems and which is the most efficient way to implement such applications. We proposed a number of test cases that can be used to evaluate power consumption and energy efficiency of virtualization systems. We run the tests on different common desktop and laptop multi-core systems. We performed the tests on several system components: CPU, memory, and I/O. In our tests we observed no significant increase in power consumption when starting and using several VMs on one PM. However an increase of approx. 1% per VM instance was measured for PM. Furthermore when running CPU and I/O intensive workloads on VM and PM we did not observe significant differences in power consumption absolute levels. But in term of energy efficiency we estimated a decrease of 5-10% when the workload was executed on VM compared to its execution on PM. This decrease in energy efficiency is mainly due to the decrease in performance.

We aim to continue the present work on investigating energy efficiency of virtualization solutions to develop an energy model for a virtual machine and use this model to build an online software

monitoring tool used by a virtual machine to estimate its energy consumption and thermal dissipation.

4 USING POWER SIGNATURES IN ELECTRIC DEVICES CHARACTERIZATION

4.1 OVERALL DESCRIPTION AND RESEARCH OBJECTIVES

Using smart grid for developing intelligent applications is a current trend of great importance. One advantage lies in the possibility of direct monitoring of all devices connected to the electrical network in order to prevent possible malfunctions and manage (optimize) the overall power consumption in office or residential buildings. Therefore, this section describes a method for an automatic detection of the malfunctioning of consumer electrical devices. Malfunctioning means any deviation of a household device from its normal operating schedule. The method is based on a comparison technique, consisting in the correlation, between the current power signature of a device and an ideal signature (the standard signature provided by the manufacturer). The first step of this method is to achieve a simplified form of power signature which keeps all the original features. Further, the signal is segmented based on the data provided by an event detection algorithm (values of the first derivatives) and each resulting component is approximated using a regression function. The final step consists of an analysis based on the correlation between the computed regression coefficients and the coefficients of the standard signal. Following this analysis all the differences are classified as a malfunctioning of the analyzed device.

In our work we propose to monitor domestic devices in order to automatically extract their workload power profiles in order to identify bad usage habits, external conditions or aging and finally failures of these devices. The main goal is to detect and model the influence of the identified elements on the energy efficiency of the devices. In our approach we monitor the target devices in order to extract power consumption profiles of different applications, workload, external conditions or usage patterns. Next we apply pattern recognition algorithms to extract relevant behavior or patterns in the power consumption series of measurements. We call the identified patterns in the power profiles, power signatures. Based on these signatures we further try to count device specific usage parameters and to identify abnormal behavior of the device due to aging or malfunctioning. The final goal of our work is to propose an intelligent house level centralized solution to monitor, analyze, predict and control household devices in order to improve their life time usage energy efficiency.

4.2 THEORETICAL CONSIDERATIONS

When monitoring power consumption of a device or component a direct relation between power variation and component activity can be observed. A consumer device consists of several internal components connected to the same external power lines (Fig. 56). Power line's electrical parameters can be measured by the mean of a power meter. Activation of each internal component during execution of one specific task or program has an impact on the electrical parameters of the system, mainly its power consumption. If two or more components are overlapping their execution the overall impact at the system level power consumption is an addition of individual contributions.

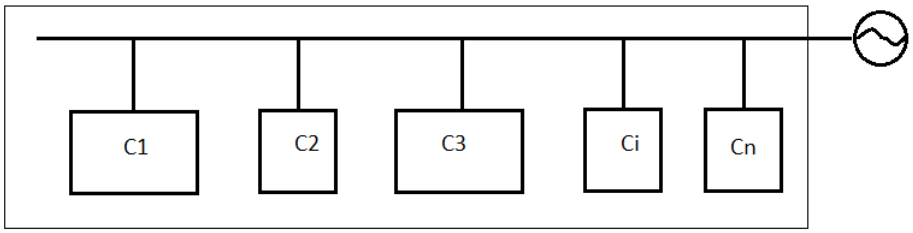

Fig. 56 System powered components

Power signature of a device is defined as the power consumption response to certain workload executed by the device. In other words a power signature is the variation in time of power consumption of the device under measurement when it executes a program or task. Power signature is specific to a device and to a workload executed by the device. Our goal is to use system level power signatures to identify component level activity while accounting for their energy. Furthermore, our initial assumption was that based on a reference signature of a working system, faulty or aged components of a consumer device can be identified and then replaced.

Power factor is a positive electrical parameter and subunit size that reflects the efficiency of electrical equipment fed with alternating current. Therefore the energy which flow across the device which is visible as apparent power symbolized S [VA], and the flow of energy actually used for productive purpose equipment is symbolized active power P [W].

The value of the power factor is given in the equation $\cos(\varphi) = \frac{P}{S}$ (Eq. 17):

$$\cos(\varphi) = \frac{P}{S} \qquad \textit{(Eq. 17)}$$

Additional to these, there here is also so called reactive power due to the reactive component (mainly inductive) consumer, symbolized by Q [VAR] and also distorting power (D) of the consumer due to nonlinearities (mainly due to the rectifier circuit).

Mentioned above parameters satisfy the relation ($S^2 = P^2 + Q^2 + D^2$ (Eq. 18).

$$S^2 = P^2 + Q^2 + D^2 \qquad \text{(Eq. 18)}$$

For improved analysis in this paper, the power factor was selected along with active power, both of them being indicated by the measuring equipment.

When analyzing more consumers electrical interconnected, dynamic consumer program will be characterized by the following parameters: power, power factor and speed variation thereof.

Active power and power factor of the system are given in relations ($P = \sum_{i=1}^{n} P_i$ (Eq. 19) and ($\cos(\varphi) = \frac{P}{\sqrt{\left(\sum_{i=1}^{n} P_i \frac{\sqrt{1-\cos^2(\varphi_i)}}{\cos(\varphi_i)}\right)^2 + P^2}}$ (Eq. 20).

$$P = \sum_{i=1}^{n} P_i \quad \text{(Eq. 19)}$$

$$\cos(\varphi) = \frac{P}{\sqrt{\left(\sum_{i=1}^{n} P_i \frac{\sqrt{1-\cos^2(\varphi_i)}}{\cos(\varphi_i)}\right)^2 + P^2}} \quad \text{(Eq. 20)}$$

where P and $\cos(\varphi_i)$ are specific parameters of each interconnected equipment in the system.

For every type of device that is an independent power consumer, we can establish different power profiles (or power fingerprints), that denote the power consumption of the device for a given utilization profile (e.g. usage pattern, applied stimuli or workload). In our tests we observed that every electronic device has a specific power consumption profile.

We grouped the consumer electrical devices in three classes according with their built-in electronic intelligence or their workload complexity:

- Low-intelligence devices are considered the consumer electrical systems with no or low built-in intelligence. In this class we include refrigerators, washing machines, heating devices, air conditioning devices, etc.
- Medium-intelligence devices are considered the electronic devices containing some level of electronic control features: TV sets, radio devices, CD players, DVD players, set-top-boxes, fixed phones, etc.
- High-intelligence systems are the computing class of systems containing at least one certain type of central processing unit like microprocessors or microcontrollers. In this class of devices we consider: desktop PC, notebooks, PDAs, SmartPhones, printers, network devices, etc.

We understand by power profiles the variation in time of power consumption measurements related to usage pattern applied to the component. We define one profile per component and workload type in order to see how component power consumption changes with component's parameters when the workload is applied. Test methodology we used to extract power consumption profiles addresses the power states of target devices. Every device has at least two power states (on and off) while most devices implement more power states:

- off - the device is turned off, but it remains plugged in;

- sleep or stand-by - the device is in one of its power saving states, where it waits for certain commands to switch in active state again. In sleep states a device is not completely switched off in order to retain at certain level the last active device state or context, so that the active state can be easily activated;
- active - the device is turned on and executes its activities.

Power signature of an electronic device is defined as the power consumption response to certain workload executed by the device. In other words a power signature is the variation in time of power consumption of the device under test when certain usage pattern or execution workload is executed by the device. Power signature is specific to a device and to a workload executed by the device. The power signature of the workload is the composition of power consumption measures of each action needed to sustain the workload. A power event is defined as a change in power values of a device that identifies a specific functional event of the product (Fig. 57). A power workload is a complete set of actions a product executes in order to provide its service. Every activity or action has its own power consumption whose profile depends of the type of electrical or electronic devices that implement them.

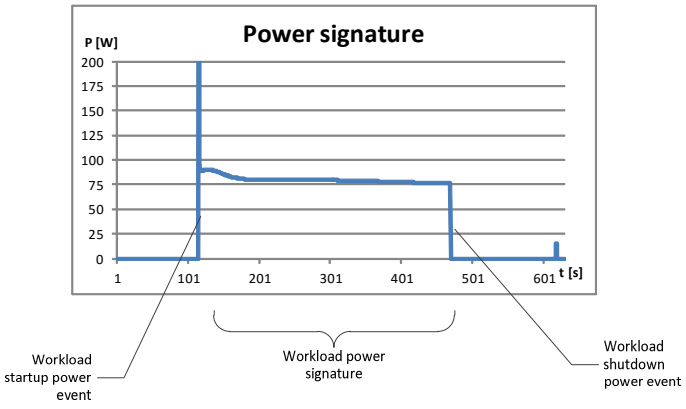

Fig. 57 Power events and power workloads

The composition of power measures of several activities are depicted in Fig. 58. Every subsystem of a device has its own power consumption. The subsystems work together to complete the execution workload. The workload power signature is the composition of power variations of the device subsystems or components activated during workload execution.

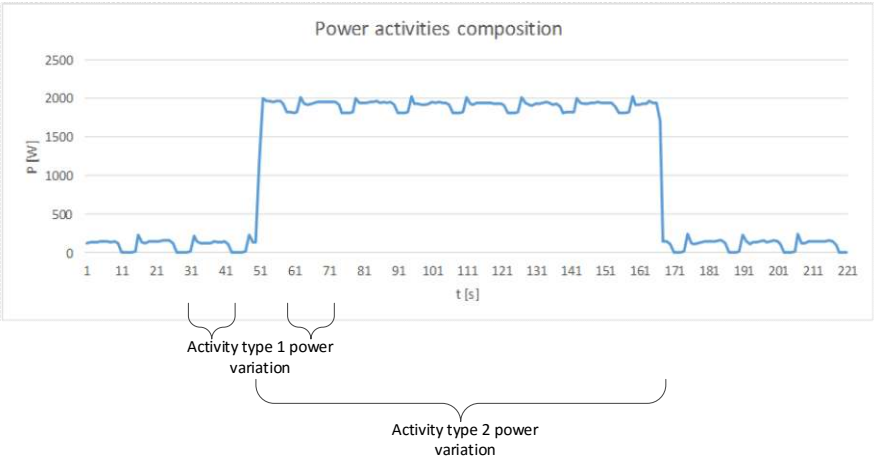

Fig. 58 Power activities composition

Obtaining the power signature of an electronic device is a simple task to achieve. We need to place a power electricity meter connected to AC lines of the electronic device under test. The intelligent power monitoring solution, presented in the next sub-section, was used for the execution of the power characterization tests. Such a test is needed to emphasize the energy consumption of one specific program or workload of the device under test.

4.3 MEASUREMENT INFRASTRUCTURE TEST BENCH

The proposed power measurement infrastructure aims to address a wide range of needs in this domain, and that was the starting principle in the design process. We have to take into account different usage scenarios in order to satisfy as many requests as possible. Also, given the use of the wireless sensors, the topology of the location has to be accounted for. That is why an incremental design process was selected, which allows for developing a base solution which serves only a small segment of clients, and then extend it with more functionality at each step. The types of devices we targeted to monitor do not vary considerably. This means that the hardware needed in the implementation of this solution is the same for all devices, which lowers the development costs.

From an architectural point of view, our system is fairly simple. There are three main components that interact in order to fulfill the purpose of the solution. These parts are: a wireless sensor network, a central unit and a web server (Fig. 60).

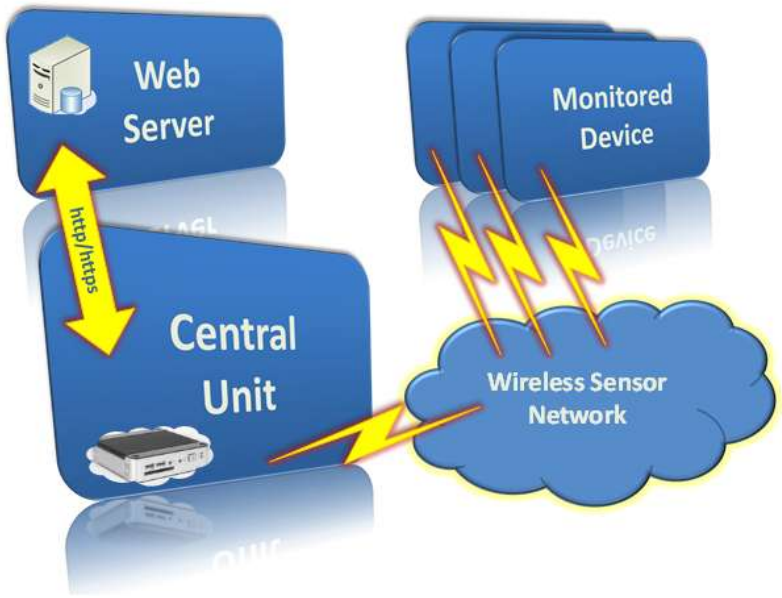

Fig. 59 Overall monitoring architecture

The sensor network is responsible for acquiring power measurements from the targeted devices and transmitting them safely to the central unit. There are two types of nodes in this network: the coordinator, which is connected to the central unit and end-points, which are connected to the monitored devices. An end-point sensor node is equipped with a wireless communication device, a power measurement device and a micro-controller which controls the activity locally.

The central unit can be implemented with an embedded device (e.g. Raspberry PI or Arduino) or a desktop PC system. The only requirements for the central node are to have serial communication port and Ethernet interface. The wireless network coordinator is attached through the serial COM port to the central unit, in order to link it to the sensor network. It is here that all measured data is gathered, analyzed and structured in order to make it easier to access. It offers a visual interface for the setup process of the system, when all the devices that need monitoring are registered with the application and the functioning parameters are set. These parameters consist of an individual identification number and name for each device, the connection parameters for the link with the data server and the client account details.

The third unit of the solution is a web server. Its main responsibilities are to store the measured data from all the clients of the service, and to make them available for analysis. In order to complete these tasks, the server has two main parts: a set of Windows Communication Foundation services for the communication with the central units and an ASP.NET application that provides the user interface. Additionally, a user account control system is implemented in order to increase security, such that not to allow one client access to another client's data.

The design of the hardware in the parts that were built by us also implied making use of the knowledge gathered from different fields. The design of the AC power measurement unit was inspired from the notions presented in electrical engineering textbooks. The DC power measurement uses electronic notions, but was simplified by using an already available shunt

monitor. The integration of the components of a sensor node into a unitary board needed hardware layout and digital logic skills. The design of the code that runs on the micro-controller is in a close bond with its hardware architecture, and the usage of different components, like the analog-digital converter, serial interfaces or operational amplifiers, requires a thorough understanding of the functioning of such devices.

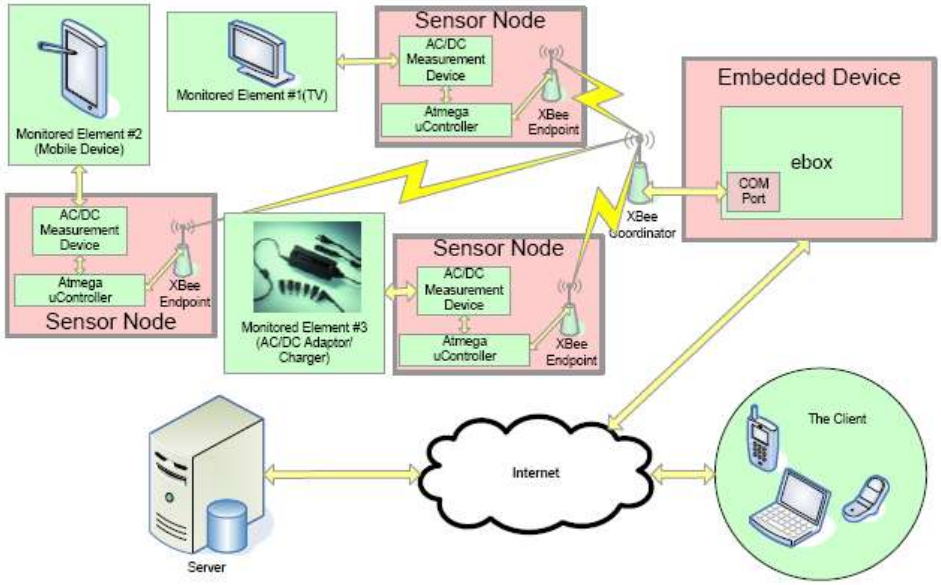

Fig. 60 Overall hardware architecture

In Fig. 60, we present the hardware architecture for the proposed solution. This design is built around the central unit (called eBox) which is considered the central embedded element of the system. Together with the sensor network coordinator, the eBox forms the central unit. The interface between the two is implemented through a serial COM port. Also, the components of a sensor node are specified in the figure. These are the XBee module, an ATmega16 micro-controller and the measuring device adapted to the type of element which needs measuring. The AC power monitor sensor node with XBee connection, presented in Fig. 61, provides a power socket with the attached monitoring and communication modules.

Fig. 61 AC sensor node

The software is structured in three separate stand-alone applications: (1) the embedded (eBox) component which can collect data from the sensors and also control them; (2) the Web portal which can be used to represent data in a logical and friendly fashion; (3) the Web Service which holds the logic to access and modify the data for outside consumers (clients).

All was centered on the data logic and modeled as a separate project with logic shared between the other components (service, reports, web portal, admin applications). As a development platform we used .NET 4 with its enhancements for Entity Framework and ASP MVC. For modeling the data layer it has been used Entity Framework 4 and Unity 2.0 in order to assure a persistence and context ignorant scenario for each application this was deployed to. Persistence ignorance was assured by POCO objects (a feature introduced in Entity Framework 4) while the context "ignorance" was handled by making use of Dependency Injection found in Unity 2.0 Framework.

On top of this came the service with its logic to access and modify database sensible content. The Service was done with WCF and if the client supports WS Binding, than it can make use of sessions (different instance/session) enabling caching and logic enhancements. The Website was meant to provide frontend business capabilities and was developed with ASP MVC 2.0 making also use of the same logic as the service. Rich client graphics and effects were accomplished by using jQuery with some of its famous controls/plugins (apple like menus, tabs, grids) which we enhanced to support different business scenarios for example paging in grids, autocomplete in combo boxes.

The software part of the system is described in Fig. 62. As mentioned before, the various components follow the general system architecture, with the code on the eBox being the one offering the main functionality. The applications written in different levels of the system have distinct characteristics and are described below. First, we first have the low-level code that is written for the microcontroller in the sensor nodes. There is a separate part that regards the coordinator. This consists of the serial communication protocol for the connection with the eBox. The code on the endpoints differs from the one written to the coordinator, because the controller on the endpoint has to communicate with the measurement device and read data from it. So, the serial interface is replaced by the interface with the devices. Both types of sensor nodes implement a communication protocol between them, which builds on top of the already existing layers. Second, we have the software on the eBox, build on top of a Windows Embedded 6.0 Image. The eBox gathers all data from the monitored devices and stores it using the compact version of SQL Server 2008 for embedded devices. A monitoring module congregates data into clusters of measurements coming from the same source and adds a timestamp to make chronological ordering possible.

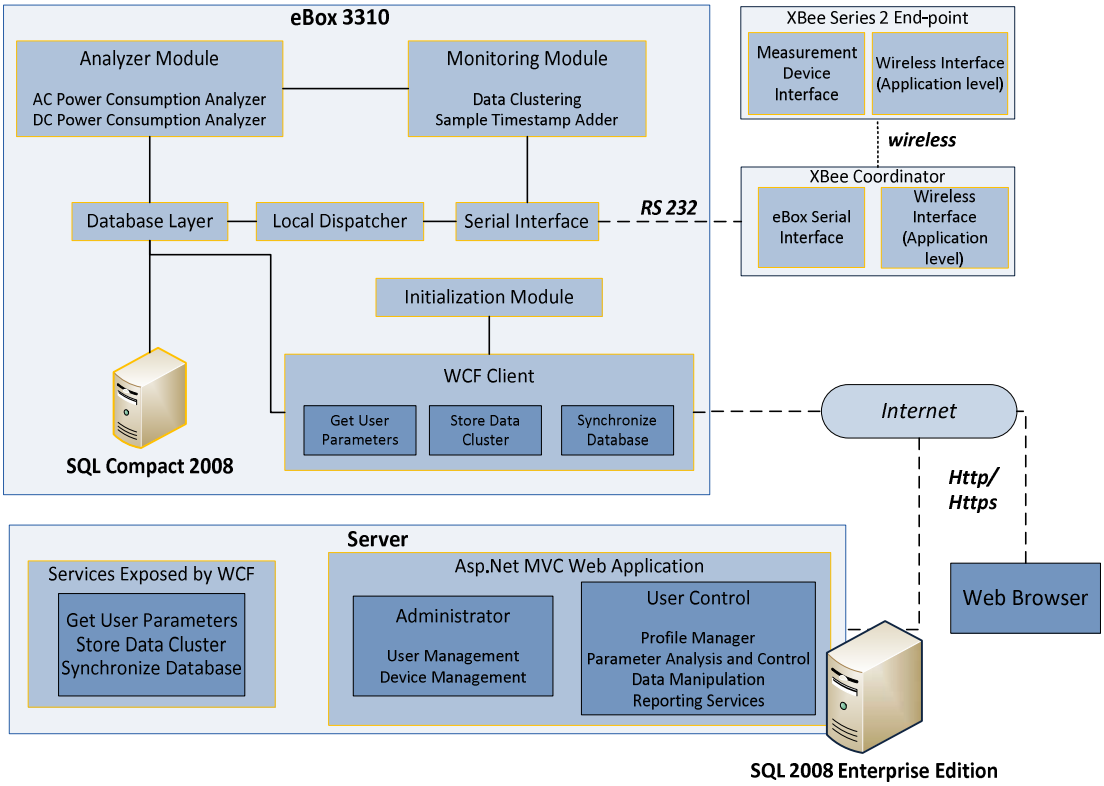

Fig. 62 Overall software architecture

Then a mild analysis is done to remove inconsistent data from the clusters, such as misreads or values that could not practically exist, but are reported due to specific events in the power line, especially in alternative current. After the analysis, the data can be stored into the database. After the analysis, the data can be stored into the database. In this state, the data is structured into chronological data samples from the same source, and each source has its own data. Reporting and analysis features were provided by Microsoft's Reporting Services 2008. Data was stored in Microsoft Sql Server 2008 R2. Though not implemented yet in the web portal, support for localization (example Bing Maps) it's present.

One important step toward our research goal is to find a simple, efficient and general method of power signature extraction and analysis in order to deeply characterize the operation of various consumer devices. The workflow of the power signature and product workload detection software application is shown in Fig. 63. Power metering infrastructure continuously measures the electrical parameters of the consumer device and stores data into the measurements database. Measurements database feeds two software components one that detects device specific events from the power series and the second that detects device specific workload. The database also contains the ideal power signature computed in advance at nominal workload parameters. A power event is defined as a change in power values of a device that identifies a specific functional event of the product. A power workload is a complete set of actions an electrical device executes in order to provide its intended service.

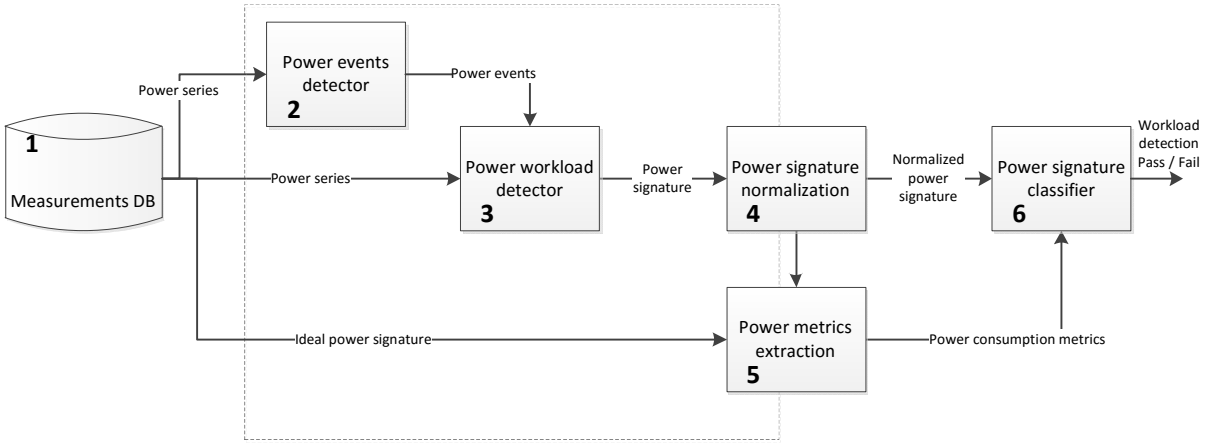

Fig. 63 Power workload detection workflow

4.4 EXPERIMENTAL RESULTS

An analysis of the entire power signature is difficult for an anomaly detection algorithm using correlation. Because of that, we decided to split the obtained shape into pieces (components) that can be analyzed more easily. Furthermore, such segmentation provides us a better localization of anomalies. Furthermore, the analysis should be personalized for each kind of electronic device, because of the differences and their specific parameters. Hence one case studies will be given in this section, addressing a laundry washing machine (medium-intelligent device).

4.4.1 Device under analysis

In our investigation we analyzed power profiles of a washing machine produced by Wirlpool (AWO/D43135) implementing the 6th sense technology. The power line electrical parameters have been measured continuously using the proposed infrastructure using a sampling rate of 1 Hz. In our tests we are interested only in static parameters therefore the sampling rate is enough for our goal.

For the purposes of this case study the system is a washing machine which we consider as an ensemble of interconnected electrical internal component, each one having a specific purpose: inlet water, water heating, clothes agitation content, water (and detergent) trays, drain water, and squeezing (Fig. 64). Water admission is carried out by means of a solenoid valve. Draining the water is carried out by means of an electric pump. Washing process is accomplished by rotating a drum driven by a DC motor fed by an electronic driver that controls speed rotation. Water heating is achieved with an electrical heater (resistance). The washing cycle is done by the washing machine controller. Of course there here are also other equipment: water level control (pressure switch) and water temperature (thermal resistance) all of them being automatically monitored by the wash machine controller. These have no observable impact on consumption them being considered as passive components. Also access door has a lock device against opening during the running of the program. This lock device is operated thermal resistance coupled to a bimetal that drives the locking device.

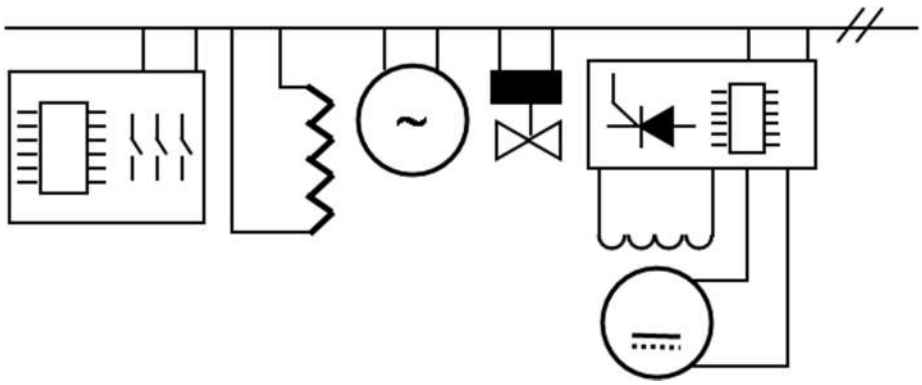

Fig. 64 Block diagram of the washing machine under test

4.4.2 Power signature analysis

Power consumption variation during execution of a washing program for synthetic laundry heated at 40° and centrifugation at 1000 rpm, is presented in Fig. 65. We selected in our tests one single type of washing program in order to detect power signature variations depending on changes in the environment (e.g. water temperature) and internal load (e.g. laundry weight and type). The power signature of one washing program is very similar from one execution to another. We can consider thus, the power signature a characteristic of one specific washing program. However, changes in power signature from one execution to another will occur due to the changes of program parameters.

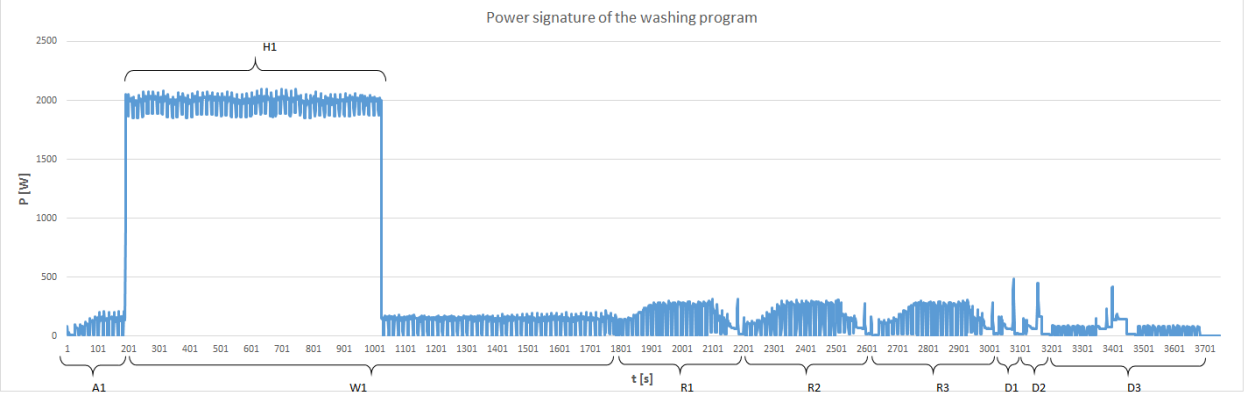

Fig. 65 Power signature of the washing program

The washing program execution phases can be easily seen in Fig. 65: startup and water admission – A1; heating (H1) and washing W1; three rinsing phases (R1, R2 and R3) and final three draining phases (D1, D2 and D3). They are further described here.

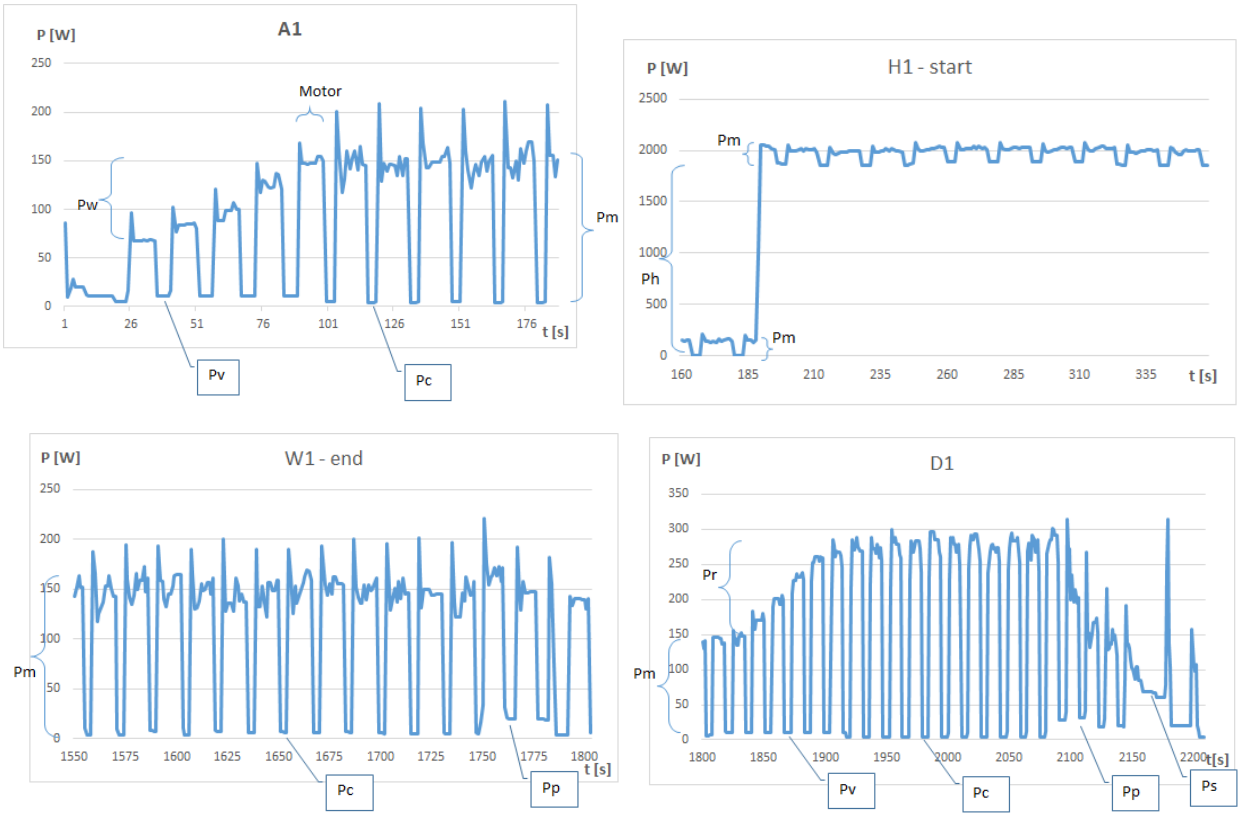

Fig. 66 Phases of the washing program**:** (a top-left) water admission signature; (b top-right) heating and washing phase; (c bottom-left) heating and washing phase; (d bottom-right) rinsing phase

Water admission phase allows water to enter into washing machine tub and soak the laundry for a while. During water admission, the admission valve is open while the engine slowly spins the washing tub. The engine activity is revealed in the power signature (Fig. 66-a) as 10 seconds long spikes occurring at regulate intervals (approx. 16 seconds). The power consumption of the engine increases steadily during water admission because of the increasing force needed to spin an increasing weight of water and soaking laundry. P_m is power consumed by the engine when spinning slowly the laundry once the admission valve has been closed. The increase in power consumption of the engine from the first activity until the admission valve is closed, P_w, is dependent on the volume of water and weight and type of the laundry. If we look at the bottom line of the power signature, two power levels can be observed: P_v and P_c. P_s is mainly the contribution of the admission valve but it includes also P_c that is the contribution of the controller board, considered constant during the washing program. Therefore, analyzing the power signature of the water admission phase we identified power consumption of three components: engine (P_m), admission valve (P_v) and controller board (P_c). P_c is a constant value in the power signature that has to be subtracted from all other components.

The main washing task takes place immediately after the water admission valve is closed. It has two main parts: heating the water and spinning the laundry within the tub. In this phase one more component becomes active: the electrical resistance used to heat the water (Fig. 66-b). P_h is power

consumption of the resistance required for heating the water. Heating H1 duration depends on the temperature of the water and target temperature required by the washing program. Subsequent heating events, smaller in duration, can also occur during this phase. The main washing phase is finished with used water draining out of the tub Fig. 66-c. P_p in the signature is power consumption of the draining pump. Analyzing power signature of this phase four components can be identified: heater (P_h), motor (P_m), evacuation pump (P_p) and controller board (P_c).

The power signature of the rinsing phase is presented in Fig. 66-d. During this phase a higher volume of water is used generating a higher power consumption (P_r). The first part the water admission valve is open while in the last part, the pump is used to empty the tub. The phase ends with high speed spinning in order to squeeze and remove the water out of laundry. Analyzing power signature of this phase several components can be identified: motor spinning with small water level inside (P_m), motor spinning with maximum rinsing water volume inside (P_m+P_r); admission valve (P_v); evacuation pump (P_p) motor high speed spinning (P_s) and controller board (P_c).

4.4.3 Ideal power signature

The first step in our analysis has been to propose and identify a normalized or ideal power signature of one specific washing program. This was achieved by using a smoothing filters and edge detection algorithms. The best results were obtained by applying a Gaussian smoothing function and first derivative edge detection. These post-processing outcomes can be seen in Fig. 67. By smoothing the periodic low speed engine activity is filtered out in order to detect automatically the main phases of the program. An event detection algorithm is necessary when we are searching for certain pieces of a signal or when we are trying to segment the signal in multiple parts. This algorithm detects all the behavioral changes that occur in signal evolution.

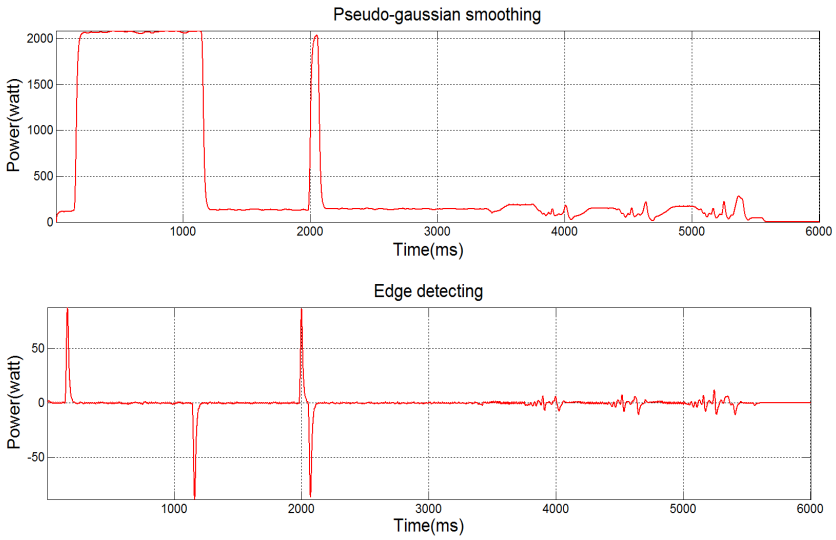

Fig. 67 Smoothing filter and edge detection results

4.4.4 Power Factor Signature Analysis

Power factor is a positive electrical parameter and subunit size that reflects the efficiency of electrical equipment fed with alternating current. Similar to power signature, power factor variation for the washing program is presented in Fig. 68. In case we draw the relation between power factor and active power for the whole program, we obtain the graphic in Fig. 10. Two clusters can be observed in Fig 10: one identifies the heating phase H1 and the second one includes components required by all other phases. These components are however hidden by the size of the first component.

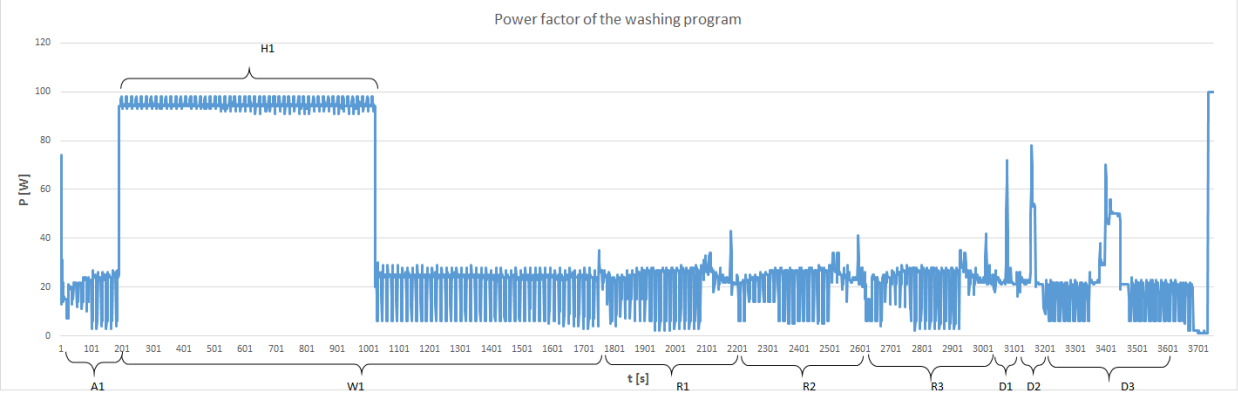

Fig. 68 Power factor signature of the washing program

In case we plot power factor and power consumption for each phase individually, components consuming less power reveals in the graphic as seen in Fig. 69for R1 rinsing phase

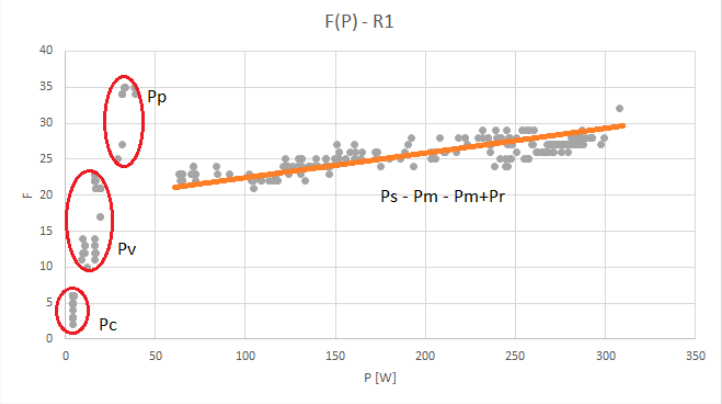

Fig. 69 Power factor and active power consumption for rinsing phase

4.4.5 Signal segmentation

Signal segmentation is done based on the signal obtained from the event detection algorithm. A simplistic approach in this case could be: any non-zero value of the signal could be seen as a change in the behavior of the device. The higher the value, the greater change has occurred in signal evolution in that moment.

This approach cannot be done in our case because the signal shows continuous oscillations and that is why most of the values are non-zero. Analyzing the values obtained from event detection

algorithm, we chose a cutting threshold that separates a neutral area (values between [-2, 2]) for an area of interest for us. Using this threshold, the important components of the signal can be seen in Fig. 70.

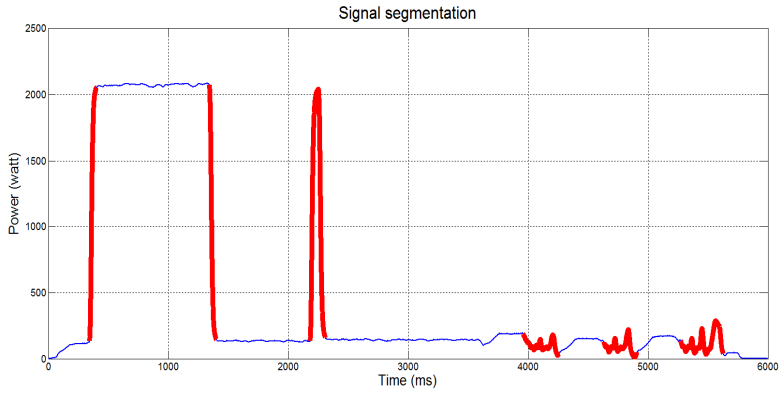

Fig. 70 Detection of signal components using an algorithm based on values from event detection method and after applying a threshold cut

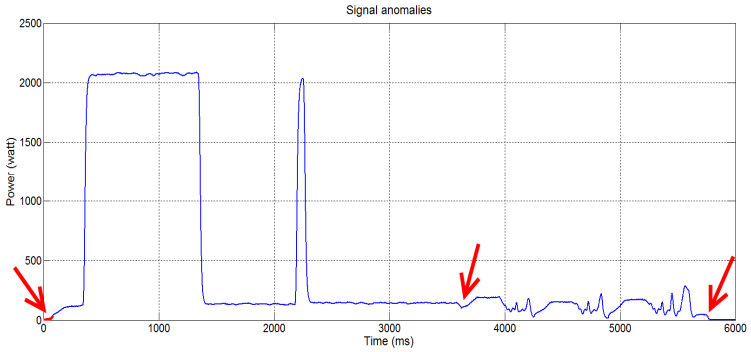

Fig. 71 Detection of short time intervals where signal has a transient behavior (anomalies)

These components succeed segmentation for a power signature signal into constituent parts. Also, the algorithm manages to detect other short time intervals too in which signal have a transient behavior (Fig. 71). It can be observed the presence of these intervals in the beginning/end of the whole program and when rinsing cycle starts. Because these intervals have a short period of time, an automated analysis of them could not be done, so they were analyzed in a separate mode.

The other proposed segmentation method is based on normalization of a signal according to another signal. For this method we need a previous processed signal (standard signal - considered to be a normal behavior of the device). This signal must have segmentation intervals defined in advance by the user. Further the normalization operation is performed in the time domain for our signal and after that, the segmentation operation is applied to time intervals obtained from the standard signal.

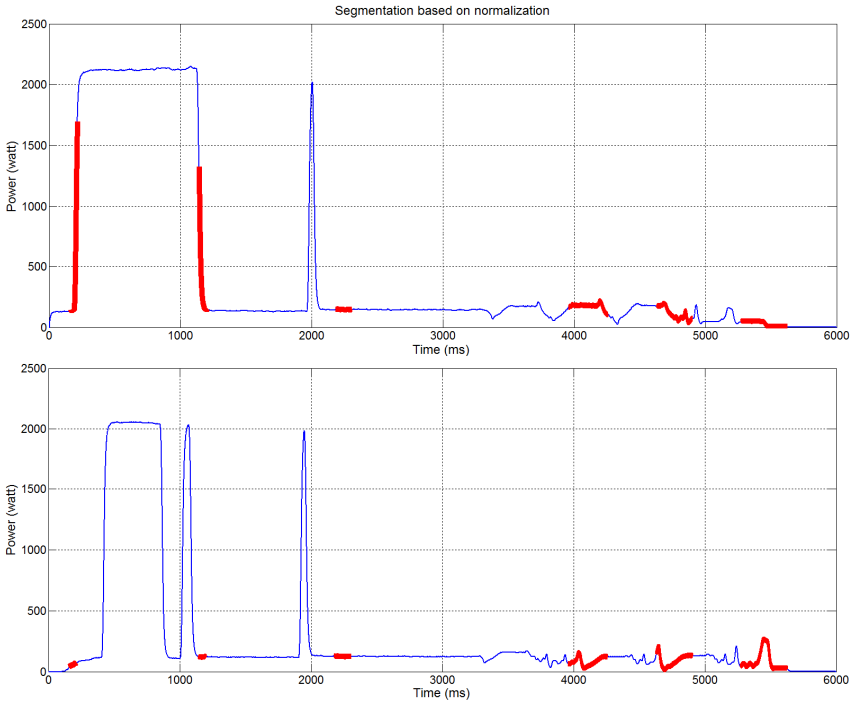

Fig. 72 Segmentation using a standard signal for other two different signals

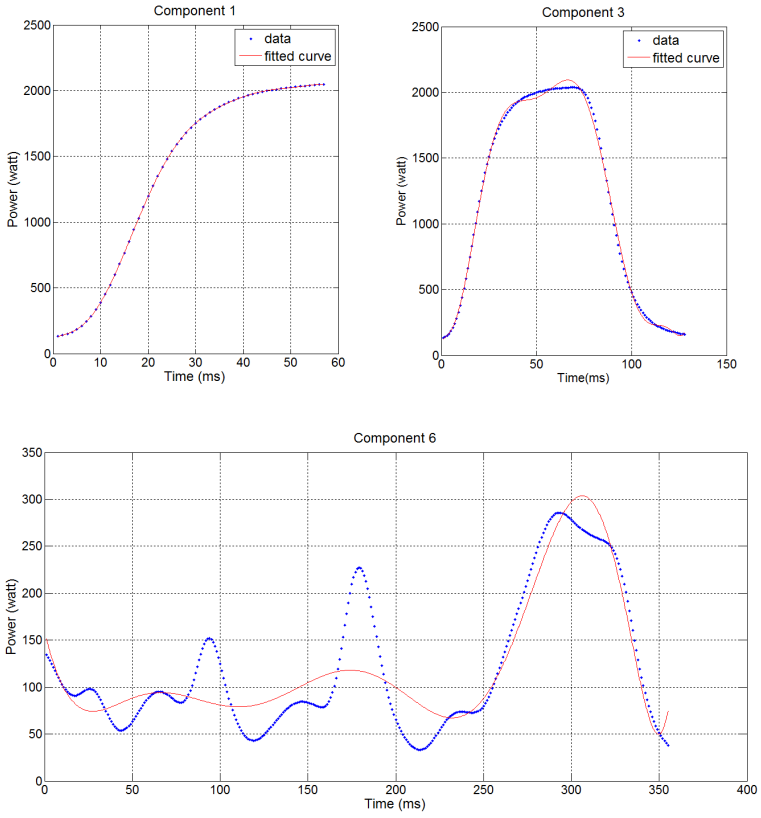

Fig. 73 Signal component reconstruction using polynomial regression

In Fig. 72 we can see the segmentation result for two signals using a standard signal. Some components have failed to be detected and for others the beginning / end are totally wrong. The results vary from signal to signal and that is why we propose this method to be used only as a secondary one.

4.4.6 Regression function for signal components

Defining a regression function for each signal component is an important part of our approach because an analysis of the signal is much easier to be done in this case. Thus, a function is characterized by a reduced set of values (parameters) than a relatively large number of sample signals. For our signal, we try finding regression functions from three classes: polynomial, exponential and a combination of sinuses.

For a polynomial regression variant we chose a polynomial of nine degree. Using a polynomial of degree greater than nine is difficult and leads to obtaining coefficients with values less than 10-5. Fig. 73 presents the reconstruction of some of the signal components where polynomial regression functions are used. It can be seen that some signal components cannot be exactly reconstructed using polynomial functions.

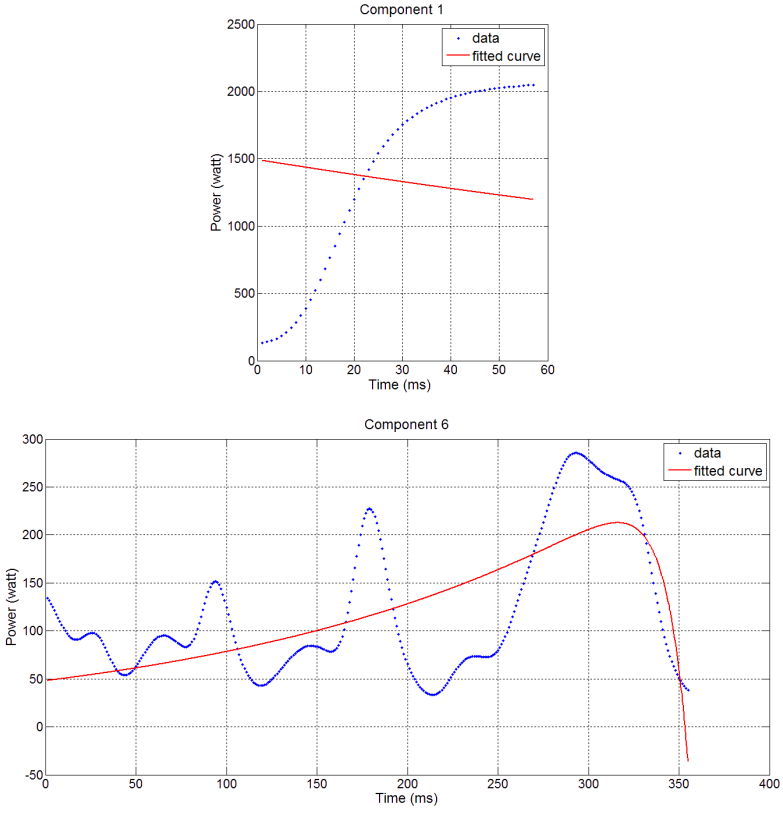

Fig. 74 Signal component reconstruction using exponential regression

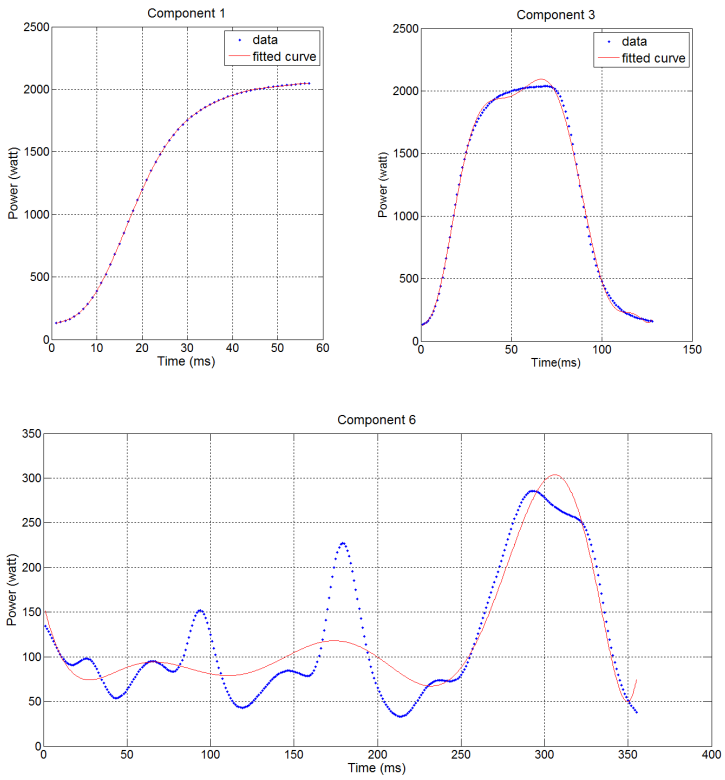

Fig. 75 Signal component reconstruction using combination of sinuses

For the second method we used multiple regression exponential functions. The obtained results for the same components of the signal are shown in Fig. 74. In this case, the components of the signal were reconstructed even worse (component 3 could not be reconstructed).

The last regression method uses a combination of sinuses (Fig. 75). It can be seen from Tab. 3 that using the combination of sinuses we obtain the best results (e.g. the smallest error). Further we used this method to find the regression coefficients for each component of the signal.

Regression method	C_1	C_2	C_3
Polynomial	0.1726	564.01	$4.6*10^6$
Exponential	358.61	776.72	$1.5*10^5$
Combination of sinuses	268.33	$3.63*10^5$	$6.7*10^7$

Tab. 3 Coefficients

4.4.7 Calculation of correlation indices between two components

Correlation is a statistical measure that determines the dependencies between two data sets and from our point of view; it has an important role as it highlights the relations of two signals. The correlation method is based on Pearson product-moment correlation coefficient and it is named "Pearson's Correlation". If we define two sets X and Y, the correlation coefficient $\rho_{X,Y}$ is given by

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y} \quad \text{(Eq. 21).}$$

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y} \qquad \textit{(Eq. 21)}$$

To calculate the correlation coefficients we need two data sets of equal length. In order to do that we should determine the signal component that has the largest length (the total number of contained points). For all the remaining components will be added (at the end) multiple zero values until we get a vector of length equal to the determined maximum length. Thus, each component will be composed of a number of samples equal to the maximum length.

Further, we present a correlation analysis based on three signals (see *Signal 1*, *Signal 2* and *Signal 3* from Fig. 76) Each of these three signals will be compared with the signal presented throughout this paper named the standard signal.

Each component of a signal will be compared with each of the components of the standard signal. A correlation index between the two components will be calculated for each comparison.

For example when we compared the *Signal 1* with the *Standard signal*, actually we compared the 7 components found in *Signal 1* with the 6 existing components in *Standard signal* (Fig. 77). The correlation matrix resulted after comparison operation can be seen in Tab. 4. We noted by SC_i , the *i* component of standard signal and C_j the *j* component of the *Signal 1*.

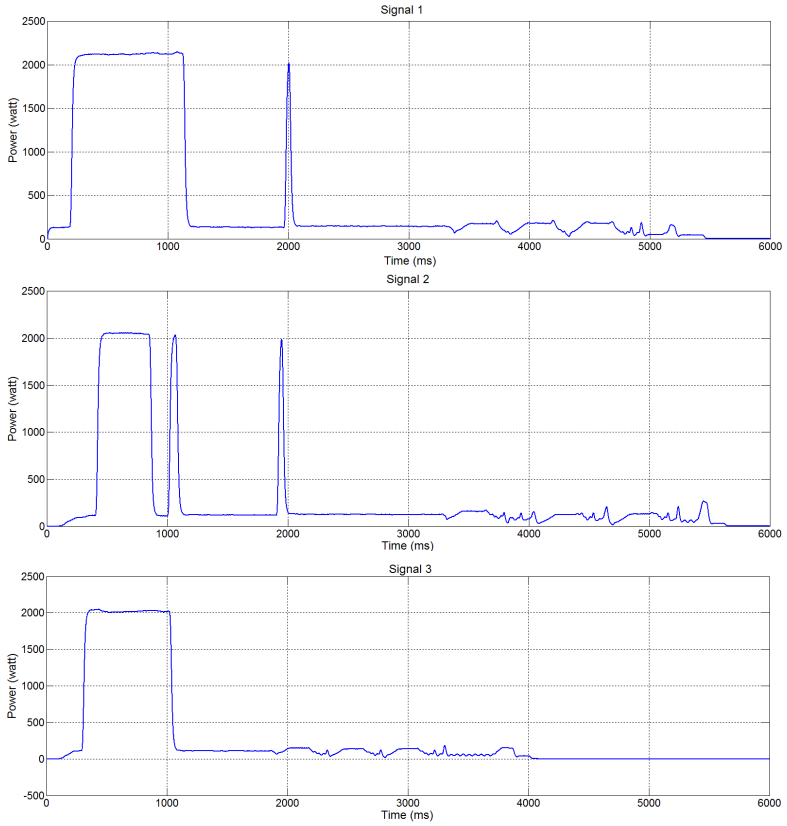

Fig. 76 Signals used in correlation analysis. From top to bottom: Signal 1, Signal 2 and Signal 3

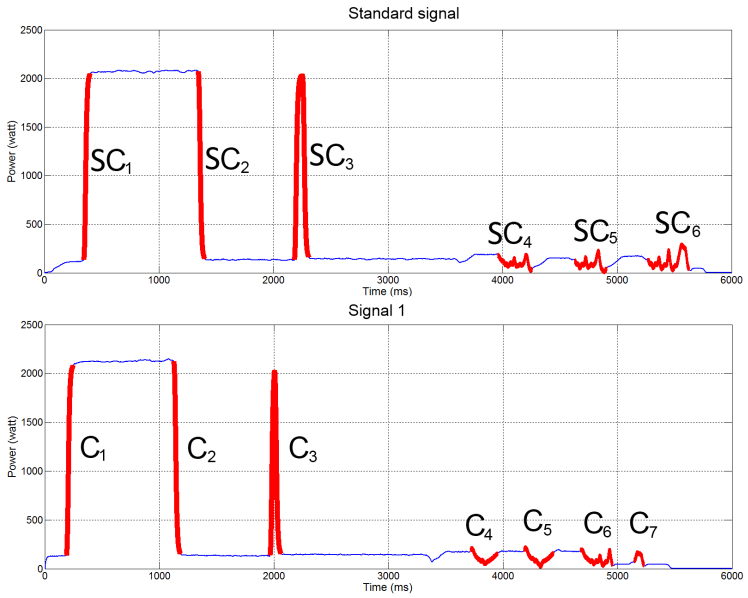

Fig. 77 Top: Standard signal contains SC_1 - SC_6 components. Bottom: Signal 1 and C_1 - C_7 components Signal 3

	SC_1	SC_2	SC_3	SC_4	SC_5	SC_6
C_1	0.983	0.315	0.685	0.330	0.030	-0.235
C_2	0.324	1.000	0.159	0.468	0.169	-0.088
C_3	0.925	0.315	0.811	0.349	0.064	-0.259
C_4	0.479	0.600	0.469	0.548	0.697	-0.490
C_5	0.489	0.611	0.483	0.741	0.526	-0.528
C_6	0.441	0.592	0.397	0.871	0.279	-0.410
C_7	0.829	0.487	0.881	0.412	0.117	-0.269

Tab. 4 Correlation matrix

4.4.8 Correlation analysis. Determining similarity and anomaly detection

It can be seen from Fig. 77 that *Signal 1* is very similar to the *Standard signal*. This is also revealed by the matrix of correlation coefficients. Thus:

- Component C_1 and C_2 from *Signal 1* are very similar with the SC_1 and SC_2 components from Standard signal (0.983 respectively 1.000). That means that the washing machine works fine in the first part of the program and does not show any anomalies.
- Component C_3 from *Signal 1* is very similar to the component SC_1 from *Standard signal* (0.952 similarity) but also with component SC_3 from *Standard signal* (0.811 similarity). This is due to the fact that the C_3 from *Signal 1* has a double length (in time) than the similar component from the *Standard signal*. Again, we can conclude that there are no anomalies present.
- Component C_4 from *Signal 1* has similarity with the component SC_5 from *Standard signal* (0.697 similarity). This means that a slightly wear of the machine is present in this part of the program.

- Component C_5 from *Signal 1* has much similarity with the component SC_5 from *Standard signal* (0.741 similarity). Again, we can mention a slightly wear of the machine for this part of the program.
- Component C_6 from *Signal 1* is very similar to the component SC_4 from *Standard signal* (0.871 similarity). Because all the SC_4, SC_5 and SC_6 are part of the rinse cycles, there is not a big issue, that this component is most similar with a component other than SC_6.
- Component C_7 from *Signal 1* is very similar to the component SC_3 from *Standard signal* (0.881 similarity). This is an anomaly because the two components are part of different washing cycles. The first is from rinse cycle and the last is from wash cycles.
- Without C_7 we have a similarity value of 0.85 between *Signal 1* and the *Standard signal*. That means a strong correlation between the two signals.

An analysis for the *Signal 2* showed a correlation of 0.823 between the two signals. It also revealed the existence of another component (similar with C_3) in this signal. This is also an anomaly and can be observed in Fig. 76.

The correlation for *Signal 3* is only 0.793, so there are many similarities between the two signals (there is a good correlation). Also the analysis detected that the C_3 for the *Standard signal* does not have a counterpart component in *Signal 3*. The absence of a standard component represents also an anomaly.

4.4.9 Synthetic analyze on components

Based on analysis done on data acquired of values for power consumption and power factor of the system, components have values of parameters given in Tab. 5. These values remains unchanged from one program execution to another. Other parameters, such as program duration, number of rotations, duration and number of heating power are variable, for the same washing program. These parameters are variable because the washing program is adaptive function on actual sensors' values.

Equipment	Active power [W]	Power factor [%]	Variability
Electric heater	1900	100	very low
Electric motor	100-500	<50	high (0.0625 Hz)
Evacuation pump	20	30	very low
Solenoid valve	10	10	very low

Tab. 5 Component level power estimation

4.5 RELATED WORK

During last decade a considerable amount of research and development work has been undertaken to improve energy efficiency of electrical products and to build an intelligent infrastructure for power distribution, monitoring and prediction. There are a large variety of research and development directions starting with energy metering devices, continuing with energy profiling of

different products and ending with various applications developed on top of intelligent power grid and metering infrastructure. In our work we direct our efforts toward energy efficiency applications based on power signatures and using the metering infrastructure we built. All this solution we call intelligent monitoring, analysis, prediction and control energy efficiency platform.

Reducing energy consumption of electrical devices has both economic and ecologic benefits, but it also opens new research directions related to the interpretation of power consumption data. The research directions we are exploring in our work is: energy and power signatures definition and their characteristics; the relation between usage patterns, power profiles and energy signature; and new application-aware power management techniques for energy efficiency. In this section we first discuss current tools for monitoring electrical consumption then we present some research papers which address power and energy monitoring and analysis.

Nowadays there are a number of options for measuring either the power consumption at the level of a whole building or the power consumption of an electric product using an on-the-self device like Kill A Watt or Watts Up Pro. In [Fehrenbacher2009] there are presented ten energy monitoring tools: EnergyHub, Tendril, Onzo, Agilewaves, Google PowerMeter, GreenBox, The Energy Detective, PowerMand, Green Energy Options, and Energy Aware. It is out of the scope of this paper to present and compare all these devices, but [Fehrenbacher2009] is a good starting point to explore these solutions.

The author of [LeBlanc2007] proposed an electrical power monitoring system containing distributed units that transmit power consumption data wirelessly via RF radios to a central base station. The designed monitoring units plug into an outlet and then the device being monitored is plugged into them. The base station parses the incoming data from multiple monitors to determine the power consumption of each device in order to have an overview on overall power consumption. Our metering solution is similar with the solution described in [Berges2009] in terms of hardware overall architectures but it much more oriented to application level to offer infrastructure for top level energy efficiency applications.

In [Adamo2007] the authors proposed to implement a virtual instrument for the electric power quality monitoring aiming to act in real-time for detecting, monitoring and recording all typical disturbances superimposed on the ideal signal. Their goal is to extract the voltage and power quality parameters for the power distribution network. The authors present a completely digital method for the fast and accurate monitoring of the electrical power quality useful to produce real time quality/ disturbance reports. In the paper, the mathematical basis of the proposed estimation algorithm is discussed in terms of reliability and uncertainty. This work is different from ours because we do not address the quality or disturbances of power lines but we consider the power signatures and their relation with the usage pattern of the device.

An interesting idea is presented in [Berges2009], where the authors try to find a way to obtain detailed information about electricity consumption in a building, at a low cost, without the usage of distinct power meter for every target device. They intend to achieve this goal using a non-intrusive method, maximizing the use of the existing infrastructure rather than imposing the need to install various new devices in the building, thus reducing the associated hardware and labor

costs. In order to split overall power consumption data they built a data acquisition system that samples voltage and current at 100 kHz and calculates real and reactive power, harmonics, and other features at 20Hz. The authors showed that under certain conditions disaggregating the total power consumption between different plugged devices is an achievable task, and can be done with a relatively high degree of accuracy. However, the problem is much more challenging in the real world, where not only does the number and type of appliances increases, but also the measurements are susceptible to more noise and obtaining ground truth data becomes more difficult [Berges2009]. Furthermore, they did not take in account complex devices, like computers, which do not have constant levels of power consumption, but it varies significantly with the running applications.

Energy efficiency and power consumption profiles are other important research topics discussed recently in the literature. In [Crosbie2008] the author explores the energy intensive behaviors of the users and how they are encouraged by the design, marketing and services supported by consumer electronics. The study based on the data collected from 20 households for TV products concludes that energy consumption of household consumer electronics is to be reduced this will necessitate the development of new products and services that are not based on redesign or incremental changes to existing products and services; but rather on providing the consumer with the function and style they require, in the most energy efficient way. McAlister et. all investigate in [McAllister2007] another growing type of devices and household energy consumption patterns: charging batteries of mobile and battery powered devices. Their study point that the large number of battery-powered devices induce a large and increasing amount of consumed energy at macro-level and about only 15% of this energy is used for battery charging, the rest is lost as waste heat during no-load and charge maintenance periods [McAllister2007].

In [Elias2009] the authors propose and demonstrate a methodology for quantifying the energy impacts of user behaviors for different residential products. They follow the product lifecycle and break down the overall power consumption of the device in several values. Firstly the user-related value which is related to inefficient use. Secondly the intrinsic value, which are the energy losses associated with the design and construction of the product, based on the intrinsic engineering technology and materials that have been used. And lastly a theoretical minimum value, which is an amount of energy that must be used in order for the product to deliver its designed function, below which it is impossible to go due to the laws of physics [Elias2009].

One of the first uses of the power signature occurred in the PCB (printed circuits boards) industry. In order to differentiate whether a PCB operates normally or exhibits abnormalities, the voltage and current values were recorded during a continuous operating mode [Miller1999]. After that, these values (power signatures) are compared with a standard shape, representing the normal operating mode for the boards. The resulted differences constitute a starting point in the decision process that specifies whether there can be operated problems with the board.

Current researches are performed with the aid of complex devices. Most of them are focused on the need of low consumption of domestic users. A recent study [Froehlich2011] provides an interesting solution for disaggregated consumption of each electrical device used. The authors manage to extract information that ultimately leads to a solution to calculate the power

consumption for each device. It should be noted that much of the research is focused on how to obtain the power signature, because the author does not aim to make an analysis process.

Starting from this, we must mention that there are numerous solutions to obtain the power signature. A new solution is proposed in [Ou2012] and uses the new concept of Internet of Things (IoT) in order to speed up the process of data acquisition. In this way, the authors ensure us that it can prevent certain natural disasters and also, the transmission of information is done with a low cost.

There are many methods for analyzing the power signature and extracting the important information from it. We start by presenting a method for classifying domestic devices using two features: the power signature and the harmonic features. [Huang2011] This study attempts to solve the problem of how to predict the power consumption especially for residential housing. For this purpose, it determines several categories of consumer and tries to classify every device in a certain category. A positive outcome of this approach lies in a better understanding of the power consumption by the residents. This article seeks to determine how a power signature is changed when there appears some malfunctioning of the electrical device. This is done by comparing the current power signature with an ideal signature (obtained by the manufacturer in the normal operation mode).

Similar goals had as well the projects described in [Lee2012] and [Jiang2009]. The research in [Lee2012] focused on two main directions: (1) the identification of a method capable to determine the load characteristics of a device and (2) several ways of signal processing needed to determine various types of workload for the analyzed device. The authors of [Lee2012] used the AC voltage and current signals and their detailed variations. In our work we used the high level digitized measured values of active power, voltage and current consumption. The authors of [Jiang2009] explain the method by which there was implemented a solution used for monitoring the AC (air conditioning) devices. Using a sensor network, it is controlling several AC devices in different types of buildings. The paper is focused on the monitoring solution presentation that is mainly used for energy usage reporting and analysis. However, some power signatures of daily usage patterns for several office and consumer devices are presented and discussed. But no automatic power signature analysis is presented in [Jiang2009].

An interesting analysis [Drif2014], based on power signature is made in order to detect the type of fault which can appear into an induction motor drive. The analysis attempts to separate the electrical faults cases from mechanical faults. Detecting faults in an electrical device, using different signatures (e.g. load signatures or frequency response analysis signatures) was also the goal for the authors from [Hassan2014] and [Abu-Siada2013].

Finding the best operation mode of the electrical devices, by studying the power signature, was also the main goal in [Jiang2009]. The authors used various sensors to monitor the air conditioning devices, and based on obtained data, they presented and discussed some usage pattern for different buildings. However, the presented solution can't be obtained into an automatic way and is not adaptable during time.

Detecting existing noise into residential power lines is another study that supports idea of monitoring house appliances through the smart grid infrastructure [Patel2007]. The authors rely on the fact that flipping a switch could produce a noise (singular or continuous) that can damage electrical appliances existing into a house.

As an earlier work we can pinpoint the work of Farinaccio et Al. that developed an algorithm able to find operating pattern of electrical appliances (each pattern was associated with an electrical device) [Farinaccio1999]. They tried to decompose the total consumption existing into a house in different parts, each part having associated an electrical device.

A similar approach was realized by the load disaggregation algorithm [Marceau2000]. By comparing a change into the total power system with each operating appliance signal, it tried to determine the home appliance responsible with that change.

Recent developments in the smart grid permit the using of phasor measurement units (PMUs) to detect possible anomalies into system [Armenia2010]. Recently, implementations of FNET (frequency monitoring network) use PMU for a better detection of anomalies [Zhang2010].

4.6 SUPPORTING GRANTS, RESEARCH TEAM AND SCIENTIFIC OUTCOMES

This work has been partially supported by the MobiPower project presented before. The intelligent power metering infrastructure has been selected two time, in 2009 and 2010 to compete at the final phase of the ImagineCup students' competition organized by Microsoft. The solution has been selected out of more than 300 projects worldwide as one of the top 15 projects.

The research team involved in the project included experienced researchers, young researchers, external advisors and students:

Eng. Catalin GHEORGHIU
PhD. Cosmin CERNAZANU-GLAVAN
PhD. Marius DARIE
PhD. Fuicu SEBASTIAN
PhD. Valentin STANGACIU
PhD. Cristina STANGACIU
PhD. Stud. Andrei STANCOVICI
PhD. Stud. Alexandru TOPIRCEANU
Eng. Bogdan POPESCU
Eng. Daniel VOLCINSCHI

There are several contributions claimed by our work:

- An intelligent power metering infrastructure deployed as a wireless sensors network for energy monitoring, analyzing, controlling and predicting of electric, electronic and computing devices;

- Definition, implementation and validation of the concept of power signature applied to run-time operation of home appliances and computer devices;
- A database storing power signatures of many kinds of devices, such as refrigerator, freezer, washing machine, bread cooking machine, TV set, DVD player, and desktop computers;
- A method using digital processing and pattern matching techniques for characterizing the functional parameters of the components, user operation modes and patterns, system aging or failures.

This section is based on the following papers published by the author: [ISI1], [ISI6], [ISI10], [ISI14], [BDI10], [BDI18], [BDI33]

4.7 CONCLUSIONS AND FUTURE WORK

In this section we present an application of this energy monitoring infrastructure to identify the device specific both intrinsic and usage parameters that have large impact on energy efficiency of these devices. The energy efficiency of studied devices is evaluated based on the patterns identified in the measured power consumption profiles, called power signatures. Based on the achieved power profiles' patterns we intend to develop power management set of control rules and notification alerts for such devices.

This section presents also a method for an automatic detection of malfunctioning for low-intelligence consumer electrical devices. The method is fully automated and it is based on the analysis of the correlation matrix between the components of the signal. It managed to analyze and to determine the degree of similarity and, to find differences between two signals. The method was tested on different power signatures obtained from 10 washing machines. During testing phase, the washing machines executed both a normal program and fractions of program. The method captures the similar phases with a normal program and prompts when abnormal behavior is presented: a phase from program is skipped or presents malfunctioning.

Also, based on similarity coefficients it is possible to compute the degree of aging. During the correlation analysis we showed how an anomaly can be captured and analyzed based on the two existing signals.

We want to further improve this method by refining the analysis part of the correlation matrix and by taking other parameters into consideration.

We are actively looking for partners interested in acquiring the innovative solution through technological transfer.

5 INDOOR POSITIONING SYSTEMS AND CONTEXT AWARE MOBILE APPLICATIONS

5.1 OVERALL DESCRIPTION AND RESEARCH OBJECTIVES

Ubiquitous mobile computing is a new and very interesting research topic with wide applicability in context-aware solutions and position-dependent provided services. One of the most significant elements of context-awareness in ubiquitous environments is mobile device localization. There have been a number of attempts to design systems for indoor localization using different wireless sensing techniques, although currently there is no conclusive method for this purpose. There are also a lot of challenges that prevent from obtaining a good positioning accuracy which are not often addressed by existing solutions.

The main goal of our work is develop a solution for indoor localization based on existing WiFi infrastructure. The positioning infrastructure is needed for delivering context-aware mobile services in indoor environments within an accuracy of several meters. This section presents the mathematical model we used to develop the positioning solution together with the set of test cases and experimental results for wireless local positioning systems based on radio signal strength in order to validate the prototype and emphasis existing issues and challenges. In our work we investigated how trilateration can be used in indoor positioning, to estimate its accuracy and its needed resources for the mobile devices. We targeted positioning in large rooms with different objects inside the room and people moving inside the room. We also want to be realistic and provide also the problems such a positioning system implementation will face during its exploitation.

5.2 THEORETICAL CONSIDERATIONS

The trilateration assumes the existence of at least three access points (APs) with known positions. The distance between each access point (AP) and the mobile device (MD) have to be determined by computations. For every distance a circle can be drawn. In the ideal case, these three circles must intersect in a single point which is actually the position of our mobile device (MD).

The method chosen for obtaining the distance between AP and MS is based on the signal strength (RSSI) measurement. The propagation of radio waves is influenced by three factors: “free space loss”, attenuation by the objects on the propagation path, and the signal’s scattering. In the absence of obstacles, the model for propagation is “free space loss” which can be expressed for the ideal isotropic antenna as in ($\frac{P_t}{P_r} = \frac{(4\pi d)}{\lambda^2} = \frac{(4\pi f d)^2}{c^2}$ (Eq. 22):

$$\frac{P_t}{P_r} = \frac{(4\pi d)}{\lambda^2} = \frac{(4\pi f d)^2}{c^2} \qquad \textit{(Eq. 22)}$$

P_t – signal power at the emitter
P_r – signal power at the receiver
d – propagation distance between emitter and receiver
λ – carrier wavelength
f – carrier frequency
c – speed of light

The ideal situation, when there's no obstacle, we consider the bi-dimensional case when all APs and the mobile device are in the same plan. If no perturbations interfered along the measuring, at moment t, M receives a signal with P_i power from the transmitter AP_i. The distance between M and AP_i could be calculated with the formula ($d_i = \sqrt{\frac{P_{ti}}{P_{ri}}}\left(\frac{c}{4\pi f}\right)$ (Eq. 23):

$$d_i = \sqrt{\frac{P_{ti}}{P_{ri}}}\left(\frac{c}{4\pi f}\right) \quad \textit{(Eq. 23)}$$

In the real world, measuring RSSI is made with some errors, because the RSSI cannot be determined with very high precision, even in the case of obstacle absence. If we take into account the attenuation induced by some obstacles the problem could be more complicated. Hence the intersection of the three circles is not a single point. For three access points the maximum number of points is six, because every two circles can generate two points of intersection.

The ideal situation, when there's no obstacle, we consider a bi-dimensional case. If no perturbations interfered along the measuring, at the t moment, M receive P1, P2 and P3 from the 3 transmitters AP1, AP2, AP3 respectively.

The receiver is at the intersection of the circles having the centre in AP_i, of radius d_i, because in every point of a circle the same power is intercepted. In ideal case, the intersection of the 3 circles is one point only, which is M (Fig. 78).

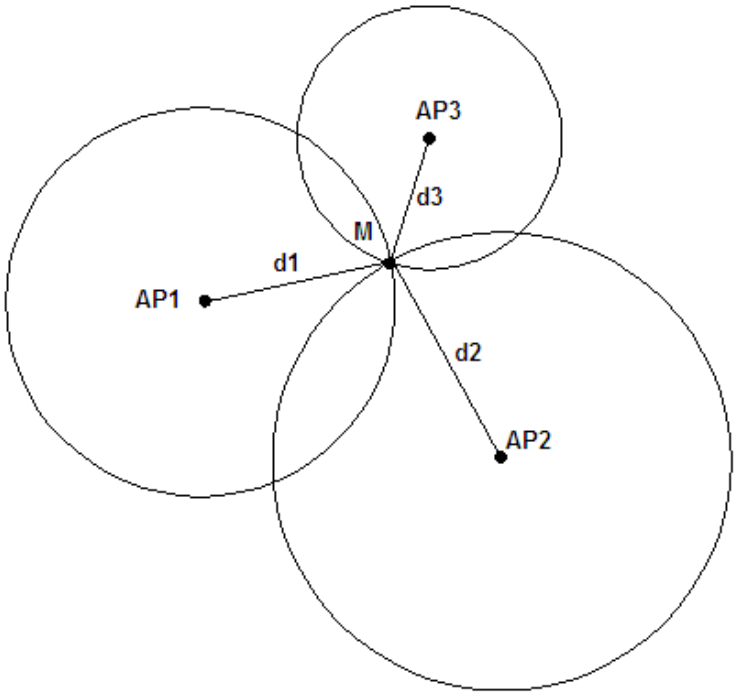

Fig. 78 Position of M in 2D ideal case

$$M:\begin{cases}\left(x_M - x_{AP_1}\right)^2 + \left(y_M - y_{AP_1}\right)^2 = d_1^2 \\ \left(x_M - x_{AP_2}\right)^2 + \left(y_M - y_{AP_2}\right)^2 = d_2^2 \\ \left(x_M - x_{AP_3}\right)^2 + \left(y_M - y_{AP_3}\right)^2 = d_3^2\end{cases} \quad \textbf{(Eq. 24)}$$

If we accept that the signal reaches the receiver influenced by some perturbations, then the intersection of the circles is no longer a circle but a domain as in the Fig. 79.

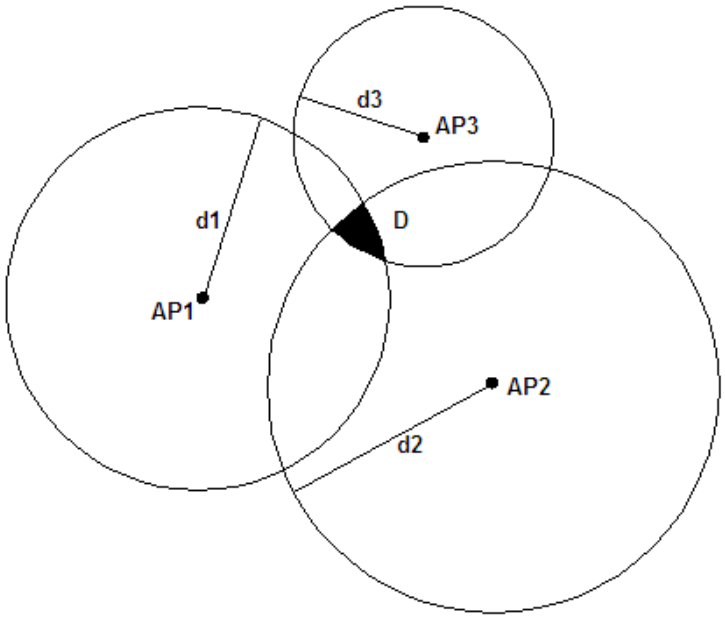

Fig. 79 The 2D real case

Domain D can be estimated to one point considering this point its centre of mass and it can be considered that the approximate position of M.

$$D:\begin{cases}\left(x-x_{AP_1}\right)^2+\left(y-y_{AP_1}\right)^2\le d_1^2\\ \left(x-x_{AP_2}\right)^2+\left(y-y_{AP_2}\right)^2\le d_2^2\\ \left(x-x_{AP_3}\right)^2+\left(y-y_{AP_3}\right)^2\le d_3^2\end{cases} \quad \textbf{(Eq. 25)}$$

The coordinates of the centre of mass are:

$$x_M=\frac{\iint_D x\rho(x,y)dxdy}{\iint_D \rho(x,y)dxdy}$$

$$y_M=\frac{\iint_D y\rho(x,y)dxdy}{\iint_D \rho(x,y)dxdy} \quad \textbf{(Eq. 26)}$$

$\rho(x,y)$ being the density of the environment in the point (x, y)

In the three-dimensional case, in the ideal variant when there are no perturbations in intercepting the signal, and the signal does not meet any obstacle, the receiver is the intersection of the spheres with the centers in AP_1, AP_2 and AP_3 of radiant d_1, d_2, and d_3 respectively, given by the ($d_i=\sqrt{\frac{P_{ti}}{P_{ri}}}\left(\frac{c}{4\pi f}\right)$ (Eq. 23) relations.

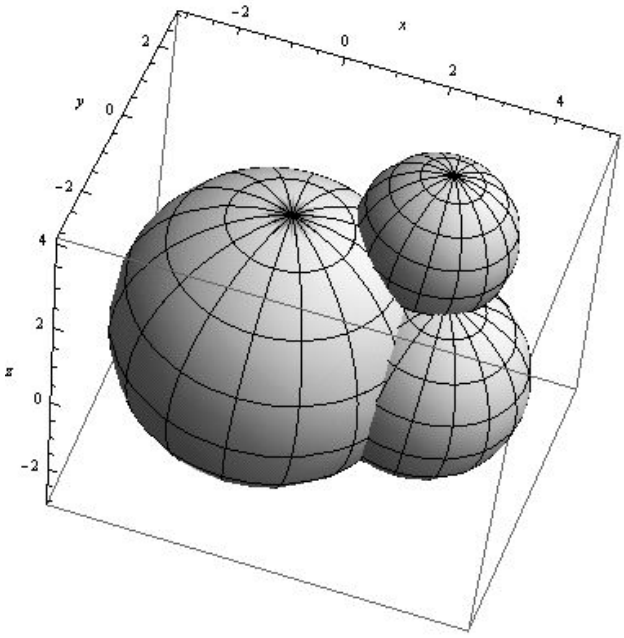

Fig. 80 The 3D real case

In the real case, where perturbations appear at the interception of the signal, the intersection of the four spheres will be a D domain which also is reduced to its centre of mass, as it is shown in:

$$D:\begin{cases}(x-x_{AP_1})^2+(y-y_{AP_1})^2+(z-z_{AP_1})^2\le d_1^2\\(x-x_{AP_2})^2+(y-y_{AP_2})^2+(z-z_{AP_2})^2\le d_2^2\\(x-x_{AP_3})^2+(y-y_{AP_3})^2+(z-z_{AP_3})^2\le d_3^2\end{cases} \quad \textit{(Eq. 27)}$$

$$x_M=\frac{\iiint_D x\rho(x,y,z)dxdydz}{\iiint_D \rho(x,y,z)dxdydz}$$

$$y_M=\frac{\iiint_D y\rho(x,y,z)dxdydz}{\iiint_D \rho(x,y,z)dxdydz}$$

$$z_M=\frac{\iiint_D z\rho(x,y,z)dxdydz}{\iiint_D \rho(x,y,z)dxdydz} \quad \textbf{(Eq. 28)}$$

$\rho(x,y,z)$ being the density of the environment in the point (x, y, z)

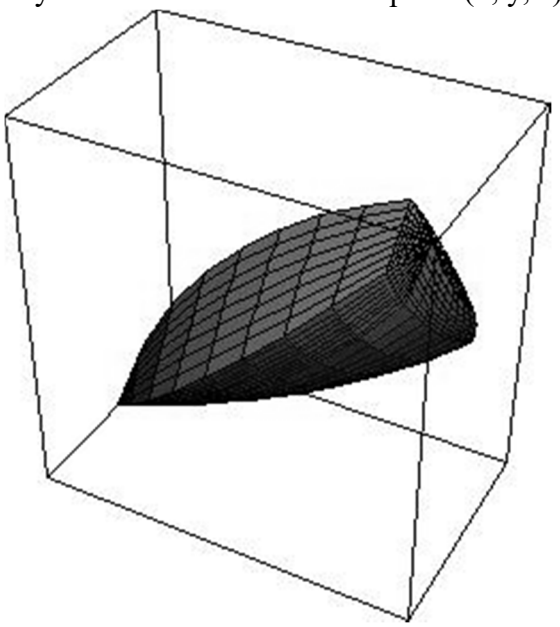

Fig. 81 D domain for the 3D case

This centre of mass approximates the position of M. In order to resolve this problem of trilateration we propose two geometric methods.

5.2.1 Method 1

In the ideal case, the solution is generated by the following system of equations:

$$M:\begin{cases}(x_M-x_1)^2+(y_M-y_1)^2=d_1^2\\(x_M-x_2)^2+(y_M-y_2)^2=d_2^2\\(x_M-x_3)^2+(y_M-y_3)^2=d_3^2\end{cases} \quad \textbf{(Eq. 29)}$$

Where, (x_M, y_M) are the mobile node's coordinates, (x_i, y_i) are the AP's coordinates and d_i is the distance between MS and AP. But, as we mentioned earlier, in a real situation, the intersection of the three circles generates not a single point, but a set of points. The problem is how to choose between those many nodes. We propose a pure geometric method, which is very simple to implement and computational efficient.

The first condition is for the APs not to be on the same axis. This condition is given by the next equation:

$$\frac{x_1-x_2}{x_3-x_2}\neq\frac{y_1-y_2}{y_3-y_2} \quad \textbf{(Eq. 30)}$$

Our solution requires making pairs with every two APs, having as a result the next relation:

$$(r_i-r_j)^2<(x_i-x_j)^2+(y_i-y_j)^2<(r_i+r_j)^2 \quad \textbf{(Eq. 31)}$$

Where (x_i, y_i) and (xj, y_j) are the coordinates of the two APs. We consider r_i and r_j the radiuses of the two circles. The radius is the same with the distance between AP and MS, distance calculated based on the signal strength sensed by de mobile device.

A. If both inequalities in relation ($(r_i-r_j)^2<(x_i-x_j)^2+(y_i-y_j)^2<(r_i+r_j)^2$ (Eq. **31**) are true, this means that we have two points of intersection. The coordinates of these points are given by the following system of equations ($\begin{cases}(x_M-x_i)^2+(y_M-y_i)^2=r_i^2\\(x_M-x_j)^2+(y_M-y_j)^2=r_j^2\end{cases}$ (Eq. **32**):

$$\begin{cases}(x_M-x_i)^2+(y_M-y_i)^2=r_i^2\\(x_M-x_j)^2+(y_M-y_j)^2=r_j^2\end{cases} \quad \textbf{(Eq. 32)}$$

$$M_1(x_{ij1}, y_{ij1})$$
$$M_2(x_{ij2}, y_{ij2}) \quad \textbf{(Eq. 33)}$$

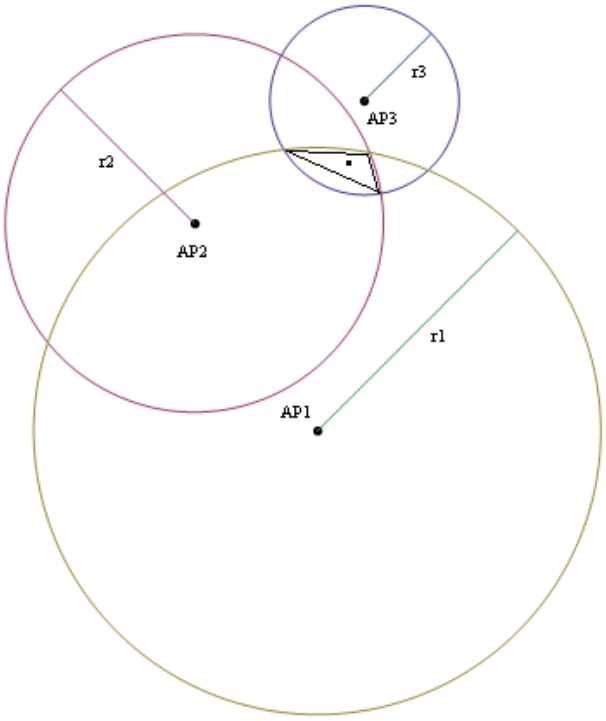

Fig. 82 Intersection points of the three circles

In case each pair of AP generates two points of intersection, it results a total number of six points, as in the Fig. 82.

B. If the right inequality of relation ($(r_i - r_j)^2 < (x_i - x_j)^2 + (y_i - y_j)^2 < (r_i + r_j)^2$ (Eq**. 31**) is false, then we have:

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2 \qquad \textbf{(Eq. 34)}$$

This means that the two circles are not intersected. In this case we consider a single point situated on the axis defined by the centers of the two circles. The coordinates of this point are:

$$M_{ij}\left(\frac{x_i + kx_j}{1+k}, \frac{y_i + ky_j}{1+k}\right) \qquad \textbf{(Eq. 35)}$$

$$k = \frac{r_i}{r_j}$$

C. If the left inequality of relation ($(r_i - r_j)^2 < (x_i - x_j)^2 + (y_i - y_j)^2 < (r_i + r_j)^2$ (Eq**. 31**) is not satisfied, then we have:

$$(x_i - x_j)^2 + (y_i - y_j)^2 < (r_i - r_j)^2 \qquad \textbf{(Eq. 36)}$$

This means that one circle is contained by the other one. As in the previous case the point is situated on the axis defined by the centers of the two circles.

$$M_{ij}\left(\frac{x_P + kx_Q}{1+k}, \frac{y_P + ky_Q}{1+k}\right) \quad \textbf{(Eq. 37)}$$

$$k = \frac{r_j}{r_i}$$

$$x_P = \frac{r_j}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}(x_j - x_i) + x_j$$

$$y_P = \frac{r_j}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}(y_j - y_i) + y_j \quad \textbf{(Eq. 38)}$$

$$x_q = \frac{r_i}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}(x_j - x_i) + x_i$$

$$y_q = \frac{r_i}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}(y_j - y_i) + y_i$$

The cases B and C generate a single point, but in case A we have two points. In order to choose between them, we apply the next computation. For each of them we make the sum of distances between the node and every other node obtained through one of the three cases: A, B or C.

$$S_1 = \sum_{k=1}^{N}(x_{ij1} - x_k)^2 + (y_{ij1} - y_k)^2$$

$$S_2 = \sum_{k=1}^{N}(x_{ij2} - x_k)^2 + (y_{ij2} - y_k)^2 \quad \textbf{(Eq. 39)}$$

If $S_1 < S_2$ then we choose node M_1, else we choose the node M_2. Thus, we generate a set of points: $M_{ij}(x_{ij}, y_{ij})$. The selected points in this way form a polygon and the position of MS will be the point M, approximated with the center of gravity of this polygon. The relations are:

$$x_M = \frac{\sum x_{ij}}{N}$$

$$y_M = \frac{\sum y_{ij}}{N} \quad \textbf{(Eq. 40)}$$

5.2.2 Method 2

The second method imposes the same general condition [14], namely in the zone where mobile device is acting to be at least three access points. If there are more than three access points a new

condition arises: the access points need to create a convex polygon like in Fig. 83. In this figure we exemplify a case with 4 access points.

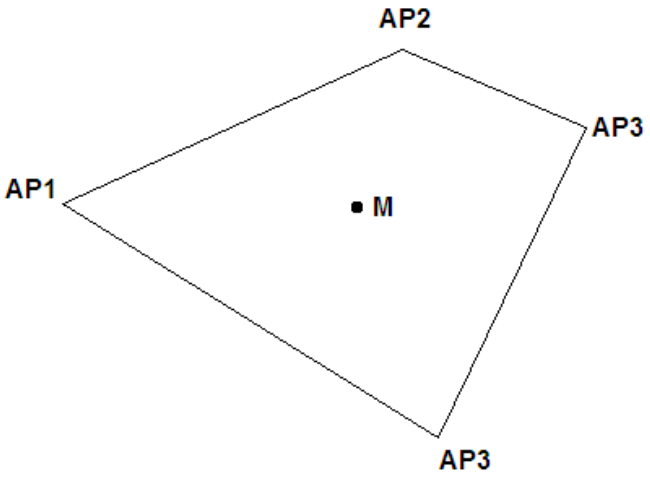

Fig. 83 Access points surrounding the mobile device

If no perturbations interfered along the measuring, at the t moment, M receive P_i power from the transmitters AP_i respectively and distances between M and AP_i could be calculated with the formula ($d_i = \sqrt{\frac{P_{ti}}{P_{ri}}}\left(\frac{c}{4\pi f}\right)$ (Eq. 23).

The algorithm of this method requires grouping every two received RSSI measures therefore we may write:

$$\begin{cases}(x-x_{APi})^2+(y-y_{APi})^2=d_i^2\\(x-x_{APi+1})^2+(y-y_{APi+1})^2=d_{i+1}^2\end{cases} \quad \textbf{(Eq. 41)}$$

Solving this system, we will obtain two solutions:

$$(x_{M1},y_{M1}) \text{ and } (x_{M2},y_{M2}) \quad \textbf{(Eq. 42)}$$

From these two solutions we shall choose the point that is at the same side of the line determined by AP_i and AP_{i+1}, with another AP (for example AP_{i-1}). We can choose any other AP (not AP_i or AP_{i+1}) because they are the vertices of a convex polygon. This point will verify the relation:

$$\left(\frac{x_M-x_i}{x_{i+1}-x_i}-\frac{y_M-y_i}{y_{i+1}-y_i}\right)\left(\frac{x_{i-1}-x_i}{x_{i+1}-x_i}-\frac{y_{i-1}-y_i}{y_{i+1}-y_i}\right)>0 \quad \textbf{(Eq. 43)}$$

The selected points in this way form a polygon. The position of point M will be approximate by the center of gravity of this polygon.

$$x_M = \frac{(x_1 + x_2 + ... + x_n)}{n}$$
$$y_M = \frac{(y_1 + y_2 + ... + y_n)}{n} \quad \textbf{(Eq. 44)}$$

5.3 EXPERIMENTAL RESULTS

In order to establish the set of test cases for evaluating wireless positioning systems (WPS), we elaborate a number of experiments, every selected experiment was performed to run a long period of time (from tens of minutes to a number of days). The test measurements have been performed at a sampling rate of 1 to 10 seconds, and saved in log files. We used two providers for wireless equipment and a number of more than 10 devices:

- Linksys: Dual-Band Wireless A+G Access Points (WAP55AG), Wireless-G Broad Band Router (WRT54GS), Dual-Band and Single-Band wireless PCI adapters (WMP54G and WMP55AG).
- D-Link: AirPlus G Access Points (DWL-G700AP) and AirPlus G Wireless router (DL-524).

The test application was implemented in Visual Studio 2005 and was run on Windows 2000, Windows XP, Windows 2003 Server and Windows Mobile 5.0 operating systems. The environment for the tests is a laboratory with two rooms (Fig. 84). The monitoring equipment are placed in different locations, according with the test case. In all graphics on the Ox axis is a representation of time and on Oy axis is RSSI in mdB. Conventional WLAN based positioning systems use radio signal strength received by the mobile terminal, in order to estimate the location of each mobile terminal separately. The local positioning evaluation site is composed of a number of access points (AP), a number of fixed or reference receivers (R) and a number of mobile receivers (M). Reference receivers are used to calibrate the system while using it. The minimum number of APs is 3. The location of the mobile stations has to be estimated based on the model introduced before.

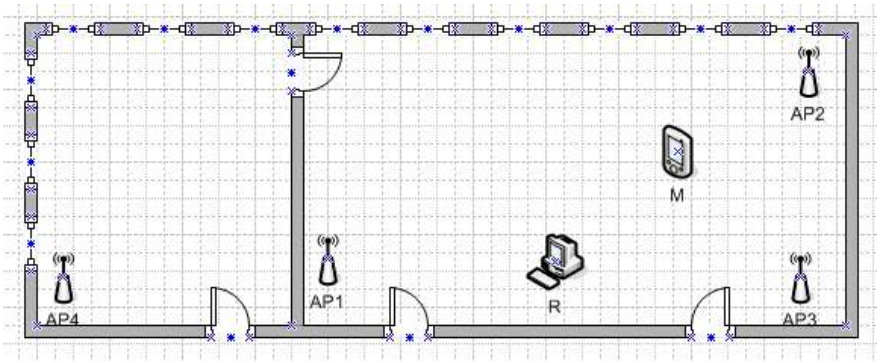

Fig. 84 WPS evaluation site

5.3.1 Wireless transmitter device types

The first test case was selected to distinguish the influence of AP device on the WPS accuracy. We used two identical APs (WAP55AG) placed in the same location in the same time and we measured RSSI in different locations in the room. This test case is intended for proper equipment selection for WPS implementation.

Experiments have shown that for different AP of the same type placed in the same location and in the same time, the RSSI measured at the receiver has different values (Fig. 8, 9 and 10). This test ran for two full days, and both 802.11g and 802.11a RSSI were logged. Differences could be observed in received signal power between the two devices, between day and night and between signal level and signal dispersion.

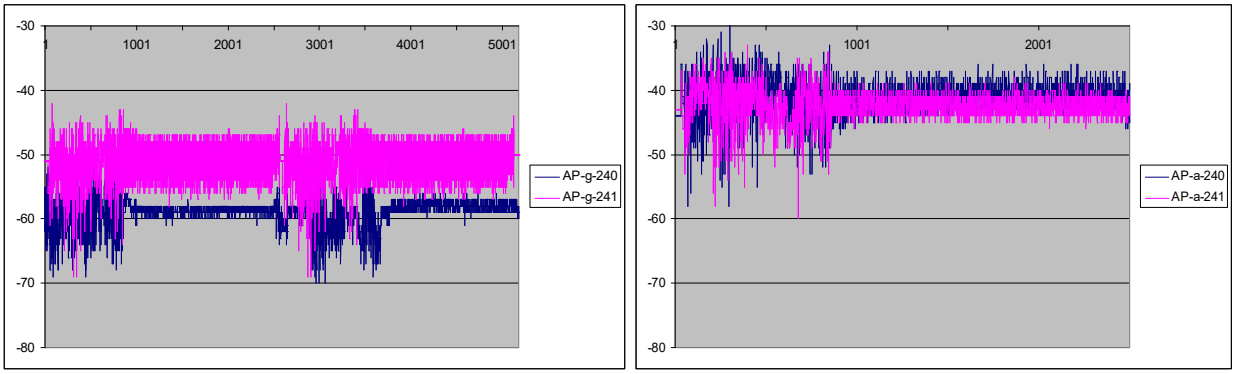

Fig. 85 RSSI from two APs placed in the same location at the same time: (a left) 802.11g standard; (b right) 802.11a standard

During the day, the room was used by the students and the presence of people has influence on the RSSI. This could be a problem because the WPS are intended to be used when the users are inside the room. Therefore a WPS should consider the day characteristics of the test when people are present and move in the room. The same result was obtained also for the 802.11a standard, but in this case the differences between devices are not so large (Fig. 85).

Averaging the measured values for both devices and standards (g and a) used we obtained the chart in Fig. 86. A difference of almost 10 mdB was measured between the received signal strength from two identical devices placed in the same location.

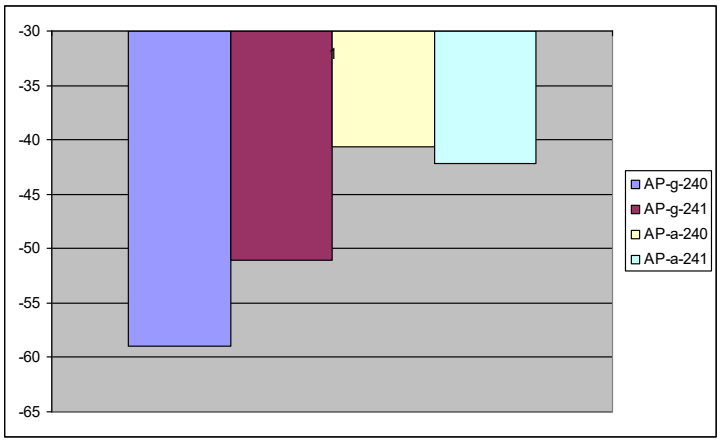

Fig. 86 Average RSSI from two APs placed in the same location at the same time

As a first conclusion we can say that before WPS deployment, an equipment selection phase is needed, in order to select the proper devices for the system. After this test case we conclude that AP-g-241 it is not recommended to be used in WPS due to its large standard deviation of the measured RSSI. A second recommendation is to use 802.11a standard in WPS or both 802.11a and 802.11g standards for a better accuracy, due to smaller deviations obtained for 802.11a in our tests.

5.3.2 Wireless receiver device types

The second test was proposed to show the influence of receiver device (hardware and software) on the RSSI. The test bench established for this test case is shown in Fig. 87, where the three different receiver devices used are placed in the same location:

- a laptop with Atheros wireless adapter integrated on the motherboard;
- a desktop with Linksys WMP55AG PCI wireless adapter;
- a desktop with Linksys WUSB55G USB wireless adapter.

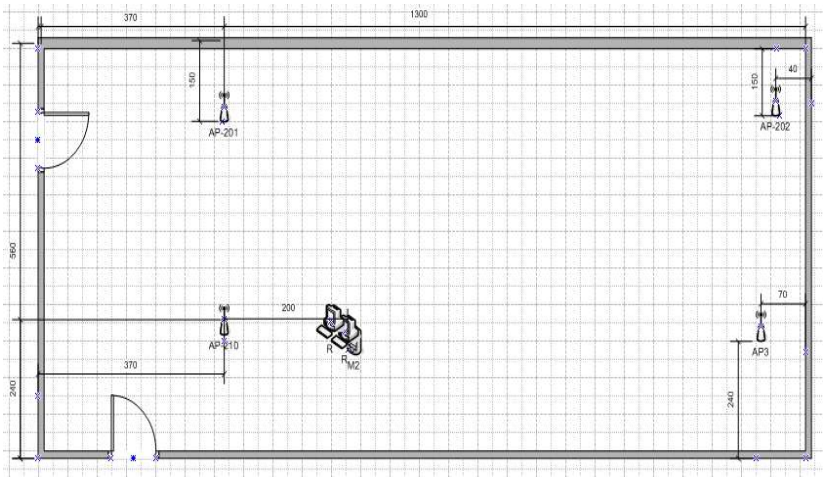

Fig. 87 Three different receiver devices placed in the same location at the same time

The average results of the measurements are presented in Fig. 88a. It can be easily observed that there are quite big differences in RSSI on the three receivers. We tested also the influences of software installed on the receiver devices: operating system, network driver, power monitoring software. The tests were run on the same receiver machine with two operating systems installations: Windows XP and Windows 2003 Server. We used two power monitor drivers and software: Wireless Zero Configuration Service from Microsoft and Linksys Monitor from Linksys. Five different access points and the results are presented in Fig. 88b.

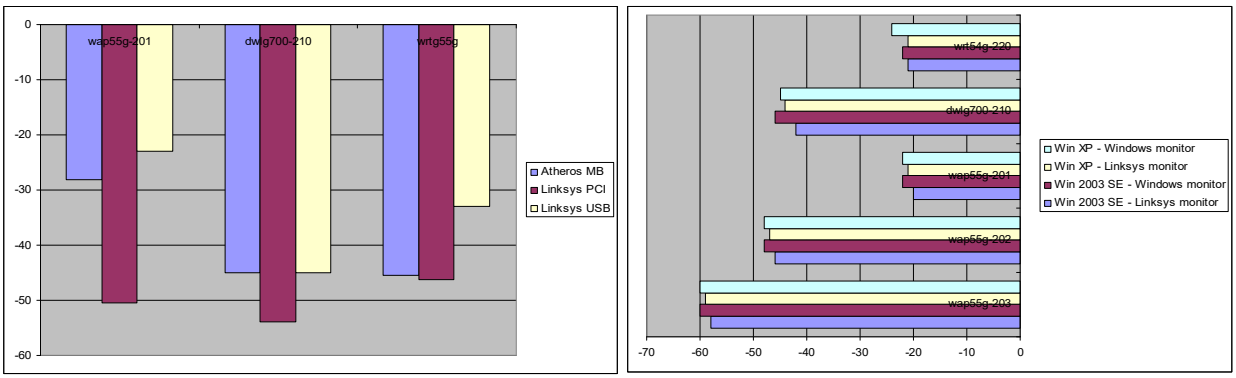

Fig. 88 RSSI measured by three different devices: (a left) network card; (b right) OS

5.3.3 Environmental factors

The third test case tried to emphasis the external environment influence over the WPS: people in the rooms, devices running time, doors opened or closed, and windows opened or closed and even the weather. Differences in RSSI measurement can be also observed, on both standards (802.11a and 802.11g), between daily hours and night hours (Fig. 85). In daily hours the room was used by people working in the lab and students, therefore the received signal has more perturbations. Better results are obtained in the night when no interferences and reflections will occur. Between the two rooms in the lab (Fig. 84) there is a door. Another test was run with the door open and then closed (Fig. 89a). Surprising in all our tests with the door closed better results were obtained.

Running the system in three successive nights, in the same conditions (device positions, door positions, etc.) some differences could be observed (Fig. 89b) in terms of average value or measurements dispersion. These differences which could be quite big, are supposed to appear due to: device running time (the number of hours from starting the device) and weather conditions (temperature, humidity, etc.)

In conclusion we showed through this second test case that a large number of uncontrolled external parameters have influence on the RSSI measurements. All these influences and parameters are very hard to be expressed in the mathematical model. Therefore a solution for this drawback is to place one or more fixed signal receivers (desktop systems – R in Fig. 84) called reference devices, in well-known positions. These devices will monitor the RSSI from all APs and will introduce a control loopback for the system.

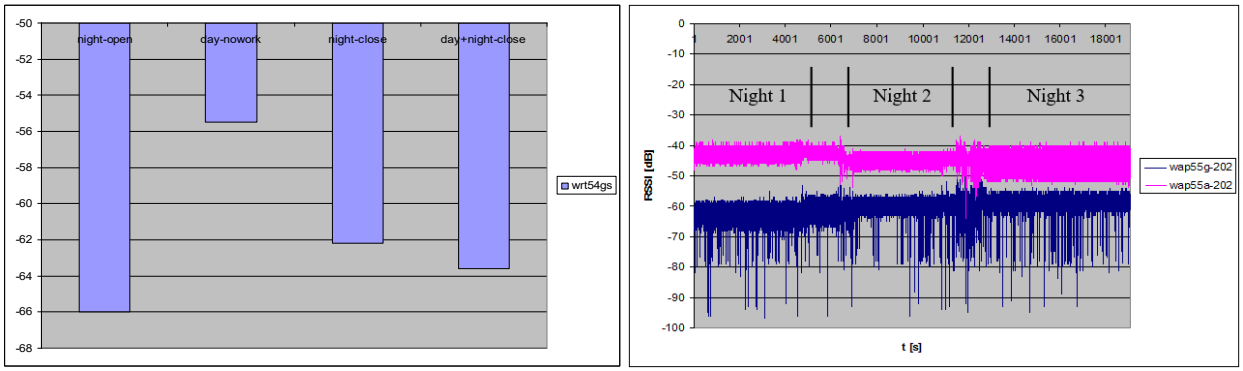

Fig. 89 Environment conditions (a left) door status; (b right) night conditions

5.3.4 Distance estimation

The main test aims at validating the relation between distance and RSSI by measurements. The test bench proposed for this test case is presented in Fig. 90. We placed a number of access points and we measured the RSSI in different well known positions. For simplicity we placed a mobile device between two APs, from 1 meter to 1 meter. For every position a 30 minutes capture was saved.

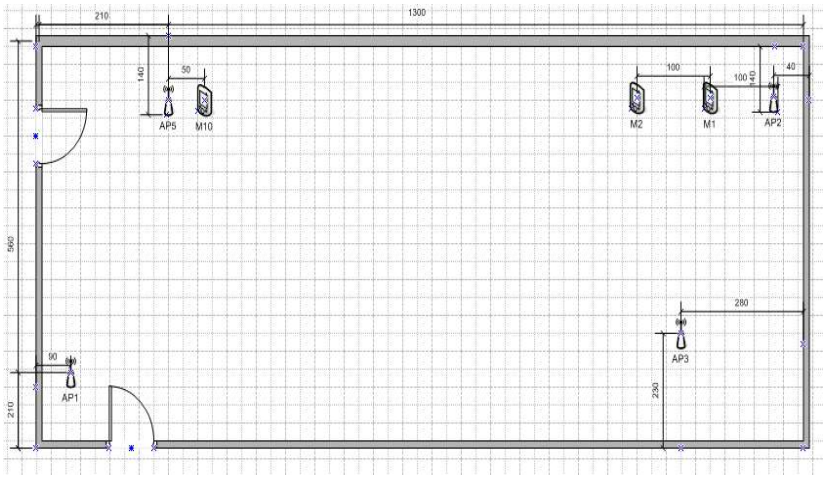

Fig. 90 Test bench for position accuracy estimation

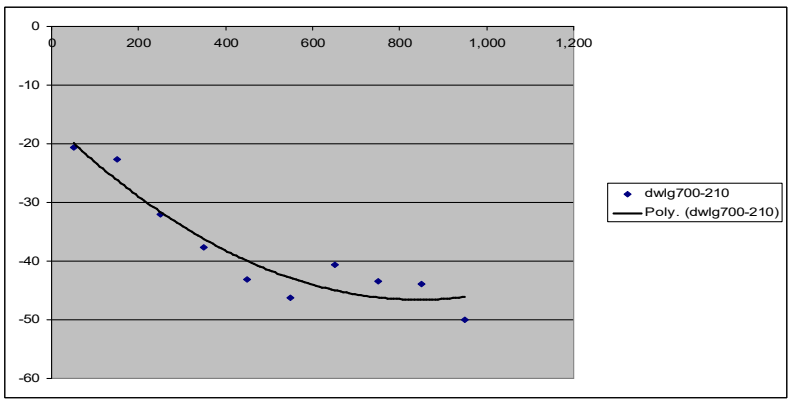

Fig. 91 Relation between RSSI and distance

The relation between the received signal strength and the distance between transmitter and receiver for some devices is presented in Fig. 91. From these measurements it can be observed that the relation between RSSI and distance are not quite the same for every device. This test set is required for the calibration of the model according to the specificity of the site where the system is deployed and equipment used. Averaging the measurements for more devices a polynomial function was obtained according with $d_i = \sqrt{\frac{P_{ti}}{P_{ri}}}\left(\frac{c}{4\pi f}\right)$ (Eq. 23). This test case could be used in a WPS implementation for transmitter signal power estimation, positioning accuracy estimation, RSSI – distance relation estimation and finally system calibration.

5.4 RELATED WORK

In the literature there are many approaches for designing methods for positioning systems, which are different in terms of distance measurement techniques and mathematical models. There are a few different localization methods that can be used for the positioning procedure. These methods are divided into three major categories based on the environment in which the information is spread: indoor, outdoor or mixed. Positioning systems consist of algorithms and methods to estimate the position of an unknown target, and are classified based on [Lim2006]:

- the signal types:
  - infrared,
  - ultrasound,
  - ultra-wideband, and
  - radio frequency
- the signal metrics
  - global location systems - outdoor methods
  - cellular location systems - mixed methods
  - indoor location systems.

The most popular outdoor localization system is known as GPS (global positioning system). The GPS receiver calculates its current position (longitude, latitude, elevation) using a trilateration technique. The distance is computed based on the time delay between the transmission and reception of the encrypted radio signal issued by the satellites. In comparison with these systems, the indoor system poses additional challenges. Different techniques exist for estimating the position of a mobile device in a wireless network [Grosmann2006]:

- cell based - with this localization technique the position of a device is simply located around the position of the access point where the device is connected.
- signal propagation time based - the position is determied using the time of arrival of the receved signal from several access points (APs) to the target device.
- signal angle of arrival - the position of a device could be estimated based on the measured angle of arrival of received signal from surrounding APs.
- receiver signal strength based - the position of a target device is estimated using the received signal strength from surrounding APs.

Cell based positioning method is very simple and considers that a mobile device is located near the known position of the AP this device is connected to. Despite its simplicity this method is not largely used because of the lack of accuracy.

The propagation time can be directly translated into distance, based on the known signal propagation speed [Broxton2005]. Based on signal propagation time from surrounding APs to the target device, its position can be estimated. These techniques offer increased accuracy but they require precise clock synchronization and more expensive infrastructure. There are two important methods in this category: ToA (Time of Arrival) [Ali2005] and TDoA (Time Difference of Arrival) [Zhang2006]. The most important parameter for accurate indoor positioning systems is the time of arrival TOA of the Direct Line of Sight path [Li2004]. Therefore an accurate estimation of TOA from received communication signals is required.

Angle measurements can be used for indoor positioning using AoA (Angle of Arrival) of the received signal or DoA (Dirrection of Arrival) [Chen2002, Priyantha2000]. For this technique additional hardware is also needed in order to measure the angle of incidence of the received signal. The AOA requires antenna arrays at each node which increase the complexity of the existing system, as well as, performing worse in multipath environment [Zhang2006].

The fourth solutions class for indoor positioning system is based on received radio signal strength of the surrounding APs. This positioning method uses the existing WLAN infrastructure and does not need any extra hardware attached to the target device. The network card in the mobile device continuously measures the RSSI (Radio Signal Strength Indicator) of the APs in its neighbor. This information is available due to beacon broadcast multiple times per second by every AP [Retscher2006]. Based on RSSI the mobile device can estimate the distance from its position to the AP it measured the RSSI. Using at least three APs with well-known positions, a mobile device can estimate its position based on measured RSSI from these APs.

RSS based WPS (Wireless Positioning Systems) were studied a lot in the last years. There are at least three location algorithms in this class of WPS:

- fingerprinting (RSS patterns [Retscher2006, Wang2005, Salter2006])
- simulation (lines of constant RSS [Grosmann2006, Grossmann2007])
- trilateration (Euclidian distance extracted from RSS – our solution)

The fingerprinting method is the most common used for WPS. An estimate of the target device position is obtained from the RSSI measurements and using a radio signal propagation model inside the building [Retscher2006]. The propagation model can be obtained using a priori RSS measurements in different locations in the building. These measurements compose a RSS map, called fingerprint, are stored in a database and can further be used to estimate the position of a target device. This method is divided in two phases [Grosmann2006]: initial calibration phase where the RSS map is achieved and the positioning phase, the measured RSS values from surrounding APs are compared to the ones stored in the database in order to estimate the current position of the device. The disadvantages of this approach are the database generation and maintenance requirements [Li2005]. During the WPS lifetime, every change in the infrastructure requires the build of the RSS map.

The second class of WPS methods uses simulation algorithms to build the propagation models of the radio signal inside the building. The simulation starts from the building map and positions of APs on the map. During the simulation for every AP the lines of constant RSS, called isobars, are computed. After the simulation the map of RSS isobars for every AP is obtained. This map can be further used by a mobile device to estimate its position inside the building. The drawbacks of this method are the complexity of the propagation model simulator and the computation resources needed to run the simulation algorithms.

A network-based localization method is presented in [Yiming2006], which uses the radio propagation signal strength that covers a 2D plane, where three sniffers are placed in order to listen to a single signal strength emitted by the mobile and to automatically generate an estimated signal strength map. Also, the algorithm is able to establish the mobile's position by browsing this table.

Unlike the previously described method a client-based localization solution is presented in [Bahl2000], using radio-frequency (RF) as well, in which the main element is recording and processing the signals received from several base stations that are placed by overlapping coverage in the 2D plane. The triangulation is achieved by using two methods: the first one requires measuring the signals and creating a signal strength map SS-MAP and searching the best signal strength measurements; the second method requires the usage of a simple propagation model in order to estimate the SS-MAP.

The authors of [Retscher2006] developed and tested a software framework called "ipos" for indoor positioning based on WLAN fingerprinting the can be used to build location based applications. It consists of an efficient, freely configurable framework, which is suitable for multiple application architectures. The RSS measurements are performed on the mobile device, the computation and visualization of position can be run on an infrastructure server or on the device itself. Based on their tests they obtained and accuracy of better than 3 meters for an area of 1500 m^2 covered by 7 APs.

The authors of [Grosmann2006] developed a method to identify intersections of lines of constant RSSI values, called isolines, of several APs within interpolated radio maps based on triangulation. Based on their tests an average deviation of 3 meters was obtained for 330 m^2 exhibition room equipped with 4 APs.

The aim of the work presented in paper [Karakos2005] is to study the possibilities that the WLAN's offer to indoor localization and to implement an application that will help to localize a device equipped with a WLAN's card supporting the IEEE 802.1lb protocol. The application developed by the authors called Wlib uses a database with signal strength values stored and a mathematical model in order to estimate the position of a user with a wireless device into a library of their University.

The disadvantage of all RSS methods is the random deviation from mean received signal strength caused by shadowing and small scale channel effect [Zhang2006]. We consider based on our tests, that the accuracies obtained by other positioning applications are the best that can be achieved and in the real life usage scenarios the positioning accuracy will decrease significantly due to: changes

in the environment, people moving around in the measured environment, mobile device speed and orientation, APs radio signal transmitter's fluctuations, etc.

5.5 SUPPORTING GRANTS, RESEARCH TEAM AND SCIENTIFIC OUTCOMES

This work has been supported by internal resources.

The research team involved in the project included researchers and students:

PhD. Marius MARCU
PhD. Fuicu SEBASTIAN
Eng. Bogdan STRATULAT

There are several contributions claimed by our work:

- A mathematical model capable of distance and position estimation of a mobile device using trilateration in a WiFi infrastructure;
- A set of test cases required to select the equipment, place the WiFi hotspots, calibrate the model and validate the WPS system deployment;
- A prototype implementation of the model able to detect mobile devices in indoor environment within the accuracy of 3m.

This section is based on the following papers published by the author: [ISI8], [ISI15], [ISI16], [ISI22], [ISI24], [BDI30], [BDI34].

5.6 CONCLUSIONS AND FUTURE WORK

A large number of hours of tests summing more than 1 month were run in order to distinguish the issues of wireless LAN-based positioning techniques and their possible solutions. Our conclusions states that in real life environment positioning accuracy is influenced by much more parameters than used in the previous articles, parameters that have to be taken in account when developing the real system. Wireless device type (both emitters and receivers), wireless communication standard, access points positions, antenna orientation, radio channel all have influence on location accuracy.

Starting from these issues of WPS we proposed a set of test cases which could be used either for WPS implementations comparing or for proper WPS configuration and deployment. The experimental results of the proposed test cases conducted us to the idea of "design for local positioning".

We finally developed the prototype including the model and using the obtained experimental measurements. We obtained a positioning accuracy of 2-3 m, which is comparable with the other work results.

This work forms the basis for retrieving context information which will be later used in extracting high-level contexts of mobile users. Mobile users have several daily or weekly contexts they use mobile applications and services. Depending on current user context the needed mobile applications may vary. Low-level context data (such as location) are collected and stored on mobile device to compose high-level user contexts. The main challenge of mobile context-aware applications development is to define the high-level context in a manner that is relevant for the user and for the applications. With respect to this challenge we propose an abstract and visual high-level context representation. For example, the user's habits reveal several static contexts (e.g. home, school and work) and several dynamic contexts (e.g. transportation). The proposed visual representation model of the revealed user contexts is depicted in Fig. 92.

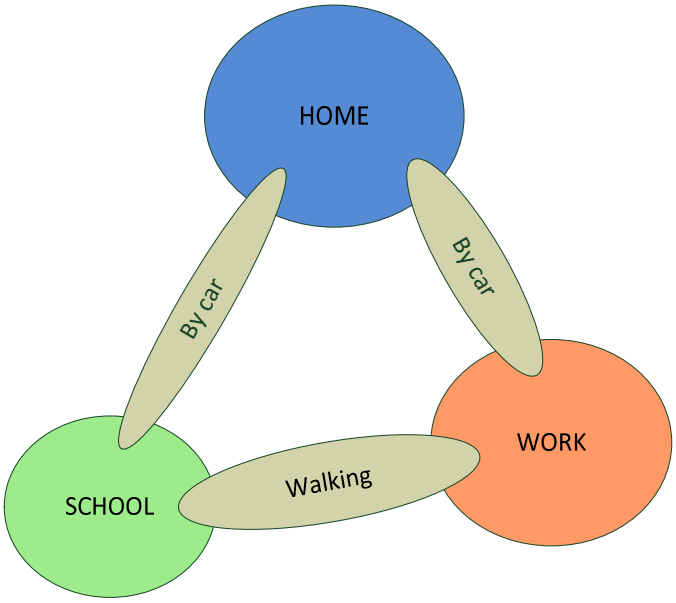

Fig. 92 High-level context abstraction

The basic elements used to construct the model are 2D ellipses. Several attributes of a context can be exposed through diverse measures of the basic elements. For example, the static context representation shown in Fig. 93 can benefit of ellipses size to describe the duration the user spent in this context and the type of wireless connectivity the user benefited within the context. The color of the basic elements describe the time of day the user attended the context (e.g. morning, afternoon, evening, night).

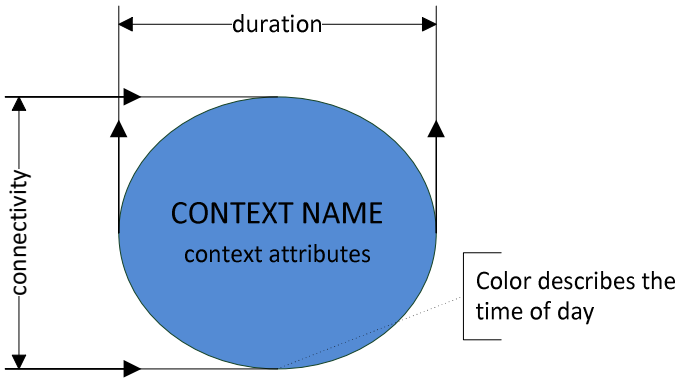

Fig. 93 Static context representation

The dynamic context representation shown in Fig. 94 can benefit too of ellipses size and color. The length and speed of movement is described by ellipse size and the color describes the type of transportation.

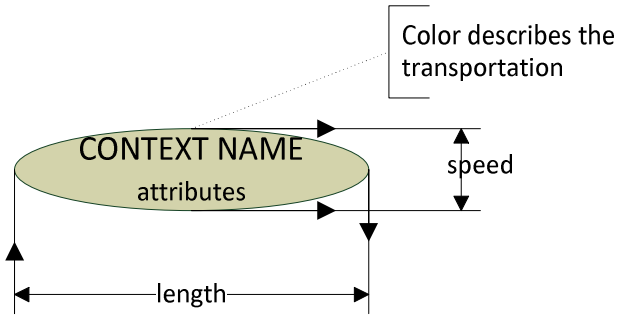

Fig. 94 Dynamic context representation

The basic context elements can be grouped together in context diagrams (Fig. 92). These diagrams describe the location of contexts (e.g. relative or absolute), placing the ellipses into the diagram according to their location.

The second challenge of implementing context-awareness in mobile applications is the process of high-level context extraction starting from available low-level context values (both current and historical). In our work we use data mining on preprocessed low-level context parameters.

Data mining represents the process of analyzing data from different perspectives and summarizing it into useful information. When it comes to using data mining for user context estimation, the main purpose would be to analyze a set of input parameters (sensors and network parameters) and see how these parameters can be classified into classes representing the user's context. Data mining usually involves four classes of tasks:

- Association rule learning which means searching for rules between the variables
- Clustering which means discovering groups of data that are similar
- Classification which means generalizing a known structure and applying it to new data
- Regression which means searching a function which models the data with the least error

6 DIRECT FPGA BASED ENERGY PROFILING OF EMBEDDED COMPONENTS

6.1 OVERALL DESCRIPTION AND RESEARCH OBJECTIVES

Power and energy profiling of multi-core embedded SoC designs is a daunting task due to the lack of fine grain accurate and high sampling rate monitoring infrastructures. Furthermore, shared components used in common by multiple processing cores, make SoC designs difficult to analyze the thread level energy consumption. FPGA development boards are currently used to implement multi-core SoC prototypes in order to validate them on real hardware and to perform thoroughly performance profiling. In this work, we investigate the capacity of FPGA designs to profile the power consumption and energy usage of multi-core embedded architectures and application benchmarks running on them.

Improving the energy efficiency of computing devices is one of the main challenges of our times. For a better evaluation of a computing device in terms of power consumption, component-level power consumption analysis has particular importance in the early stages of the design. However, accurate energy profiling can be performed only on a physical prototype, which is available during the final development stages. Instead, model-based energy profiling is usually performed, but it lacks the physical measurements accuracy and calibrations. Reconfigurable devices are being used in a wide range of applications, therefore, they are good candidates to perform physical energy profiling during prototyping.

The main goal of our work is to perform both hardware and software accurate profiling of any multi-core architecture using low level profiles of its components, such as CPU cores, memories, and buses. Therefore, in this paper, we investigate the capacity of FPGA designs to profile the power consumption and energy usage of multi-core embedded architectures and application benchmarks running on them.

6.2 FPGA BASED ENERGY PROFILING METHODOLOGY

6.2.1 Measurement setup

Performing energy profiling on a multi-core SoC is a difficult task due to the multitude of internal hardware power consumers and lack of individual per-module power rails with monitoring capability. Furthermore, some modules such as processing cores, consumes less or more energy depending on the software they execute. Energy profiling requires thus many physical measurements which must be done with high precision in a strictly controlled environment using dedicated energy profiling benchmarks.

Many FPGA development and evaluation boards provide hardware and software embedded developers with built-in support for physical measurement of current consumption. This goal is achieved by measuring the voltage drop on very small and accurate shunt resistors (milliohms) that are placed on the existing power rails powering individual FPGA modules. Accuracy of the

measurements may differ from one board to another, as there are different accuracy values used for shunt resistors, as well as for the ADC resolution and sampling rates. Main requirements of FPGA based energy profiling are high measurement resolution (at tens of uA) at high accuracy and high sampling rates (higher than 1 KHz). In case the sensor and ADC accuracy is low, high level of noise will influence the analysis. If sampling rate is low, measurements and analysis will miss accounting the short events.

We performed power and energy profiling tests on ZC702 evaluation board. It provides a hardware environment for developing and evaluating applications for Zynq 7000 SoC based on XC7Z020 device. The ZC702 board provides features required by any multicore embedded system, such as DDR3 shared memory, shared L2 cache and fast on chip memory (OCM), and general and common I/O interfaces (GPIO, UART, USB, Ethernet). The Zynq 7000 SoC consists of one integrated processing system (PS) and programmable logic (PL) on a single die. The PS includes two ARM Cortex-A9 cores, AMBA AXI interconnects, internal memories (L2 cache and OCM) and peripherals. Both ARM cores run at 667 MHz and contain local L1 data and instruction caches. The PL consists of a 7-series FPGA core, featuring 53200 slice LUTs, 106400 slice registers, 140 Block RAMs and 220 DSPs. PS and PL running independently are interconnected through several AXI based high performance interconnects and general I/O ports. [ZC702]

The ZC702 board includes power management and monitoring infrastructure. Voltage and current monitoring and control are available for selected power rails through three onboard power controllers interconnected through one PMBus [ZC702]. Out of the ten power rails that can be measured independently, the following one are of interest in our investigation:

- VCCINT is a nominal 1.0V supply that powers all internal PL FPGA core. The power consumption of the target design implemented by FPGA logic can be measured as a whole on this power line.
- VCCPINT is a nominal 1.0V supply that powers the all PS internal components including ARM cores and internal caches and OCM.
- VCCBRAM is a nominal 1.0V supply that powers all the Block RAM memories of PL. BRAM transactions can be analyzed independently by FPGA core circuits.
- VCC1V5 is a 1.5V nominal supply that powers several board components including the DDR3 memory. DRAM memory transaction performed from PS or PL can thus be analyzed from energy perspective.

In conclusion, power consumption of PS, PL, BRAM and DRAM can be measured individually, while power consumption of PS and PL internal components have to be estimated out of global measurement and component usage. ZC702 power monitors sample measurements at a rate of 10 KHz and 2 mV measurements resolution.

6.2.2 Measurement achievement

The same multi-core design can be implemented either with FPGA or ASIC technologies. The comparison between FPGA and ASIC in terms of power consumption is problematic due to the heuristic nature of the design synthesis. On FPGA, a full logical block might be used for one single bit register, which is a single flip-flop in ASIC. However, if the FPGA logical blocks fits the design

very well, it translates to more complex combinational logic and flip-flops on an ASIC. Therefore, there is a high degree of uncertainty in terms of power, area and timing.

The power consumed inside an FPGA is divided in two major categories: static power and dynamic power. The static power is dissipated whenever the FPGA board is connected to a power-supply, the design is programmed on FPGA and before applying any clocking. On the other hand, dynamic power consumption is highly correlated with the gate level switching activity, which is utilized by existing solutions for power estimation models. Therefore, dynamic power consumption measurements is worth investigating in order to study the design behavior in time and energy using hardware emulated implementations.

The main goal of our work is the investigation dynamic power consumption behavior of a multi-core design implemented in FPGA technologies by direct measurements.

The dynamic power has two components: the dynamic-idle power, which dissipates whenever the FPGA core is clocked and the dynamic-active power, which is dissipated according to the core utilization, the last one being larger and harder to estimate. Thus, the dynamic power can be measured by making modifications to clock frequency, voltage supply values, activity rates and usage patterns. As the input vectors determine the activity of the circuit, changing them can help measure the dynamic power.

Idle and workload activity have to be defined for every component to profile (Fig. 10) in order to measure and highlight the dynamic power. Different workloads proposed in the next subsection are defined by specific hardware test benches and software benchmarks. Every test is carried out in three phases while power measurements are recorded: pre-workload measurements (idle), workload activity measurements and post-workload measurements (Fig. 10). The energy accounted to a benchmark or workload is obtained by averaging power consumption measured during workload activity and subtracting the idle activity average power, the result multiplied by workload duration. Considering that the complex workloads are split into simple basic operations repeated for sufficient time, the dynamic energy can be estimated and accounted.

In order to measure and estimate energy consumption of the platform implemented by FPGA fabric and development board, its power consumption variations have to be analyzed. Power profiling of host FPGA development board aims at monitoring power consumption variations of development board components, including FPGA core, with respect to FPGA workload activities. The tests were conducted simultaneously on two identical Zynq boards. The temperature in the room was maintained at 26 degree Celsius using an internal air conditioning installation. For all test cases we performed three series of measurements. A series of measurements means that we load into FPGA a certain configuration and, after we reset the board, we monitor the power lines of the entire board according to the measurement phased described before (Fig. 10). Each test has been repeated three times with the same configuration. Thus, for each test case we obtained 6 time series.

6.2.3 Hardware Energy Profiling Test Benches

The first set of tests have been performed using different numbers of counters powered at the same voltage and clock frequency (system clock) as described in Fig. 95. For this goal a different number

of counters were synthesized, starting with one counter and ending with the maximum number of counters that fit the FPGA core. Each counter has 32 bits. The first set of tests aimed at measuring power consumption of the LUTs blocks of FPGA fabric. This test set is expected to show a linear relation between the FPGA used area by the counters and FPGA core power consumption.

The second set of tests were performed using a phase-locked loop (PLL) circuit, which allows us to multiply/divide the clock frequency. During the second set of tests, we changed the operating frequency for the counters as follows from 10 MHz to 250 MHz. This test set is expected to show a linear relation between the counters' switching frequency and FPGA core power consumption.

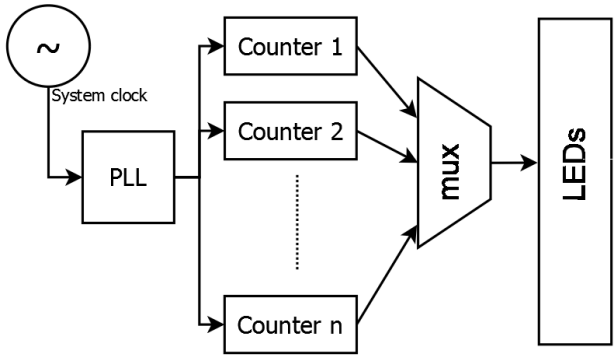

Fig. 95 Area/Frequency test bench

Block RAM memories test form the third test set. BRAM can be profiled independently while it is stimulated by RD, WR and RD+WR sequences implemented by a memory test bench programmed into FPGA core. For profiling several pairs of test bench – BRAM have been used, starting with 1 and ending with 32 as presented in Fig. 96.

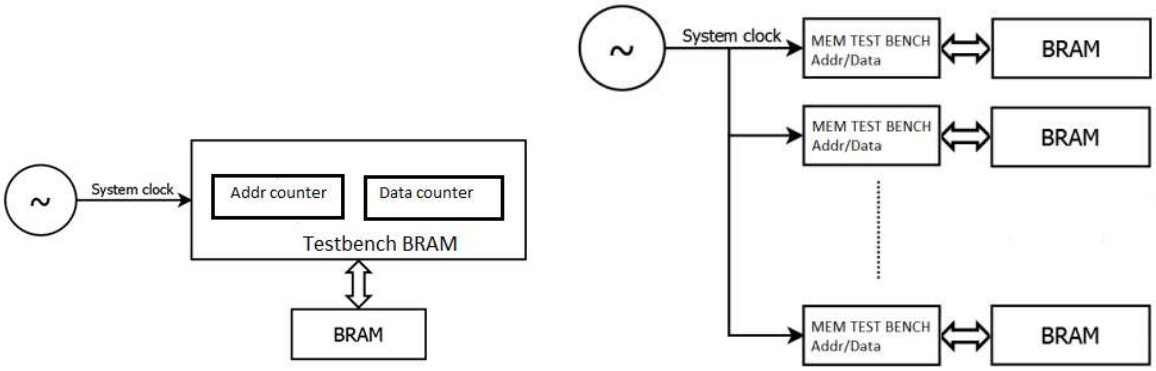

Fig. 96 BRAM test bench

In the fourth set of tests, we implemented FIR filter processing blocks (Fig. 97) whose parameters and values are taken from BRAM memory. With this set of tests the DSP blocks and LUTs logic have been evaluated from power consumption perspective.

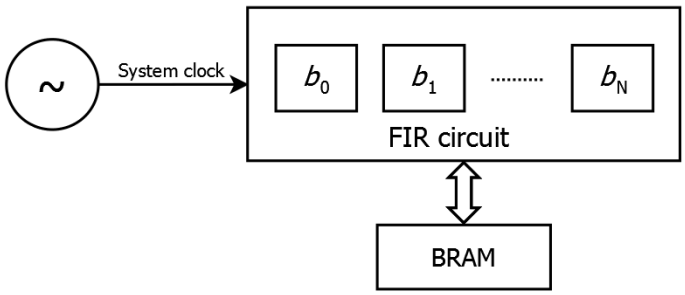

Fig. 97 DSP test bench

6.2.4 Software Energy Profiling Test Benches

More complex hardware components such as processing cores and DRAM controller and memory have to be profiled at software level. Two test cases have been performed in order to estimate power consumption and energy consumption of load/store operations on DRAM (Fig. 98): (1) execute memory transfers form a RISC core (Microblaze) programmed in FPGA and (2) execute memory transfers from ARM cores. For each test case, several benchmarks have been performed:

- sequential loads (LDR-seq) – blocks of read operations from successive addresses;
- fixed address loads (LDR-fix) – blocks of read operations from the same address;
- sequential stores (STR-seq) – blocks of write operations to successive addresses;
- and fixed address stores (STR-fix) – blocks of write operations to the same memory address;
- blocks of RD/WR operations.

The memory transfers blocks sizes have been varied for each test.

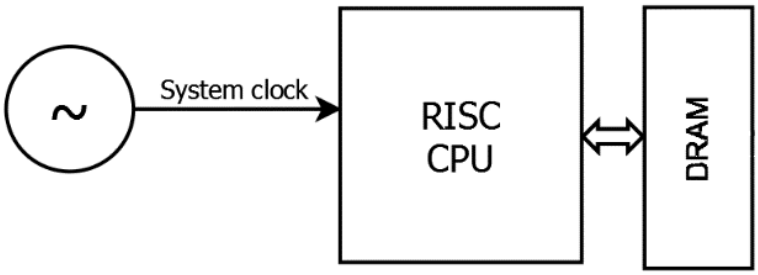

Fig. 98 DRAM test bench

Profiling a processing core is achieved by profiling each instruction individually. For each CPU instruction, a program called micro-benchmark, has been written in order to capture the power consumption variation induced by the instruction. The micro-benchmark consists of two main loops: one loop which contains nop instruction and one loop which contains the instruction to profile (Fig. 99). Microblaze is a 32 bits soft core embedded RISC processor provided by Xilinx is optimized for implementation in FPGA. It is highly configurable with instruction and data L1 cache, debugger, performance counters or floating point unit.

```
main_loop:
  xor   R2,R2
  lw    R1,R2
  add   R2,1
loop_nop:
  nop
  nop
  nop
  ....
  b loop_nop
loop_instruction:
  add   R3,R8
  add   R4,R9
  add   R5,R10
  add   R11,R6
  ....
  b loop_instruction
```

Fig. 99 Instruction profiling micro-benchmark

6.2.5 System Level Energy Profiling Benchmarks

The reference system design multi-core architecture energy profiling is presented in Fig. 100. It has one clusters of four Microblaze soft cores. The clock frequency for the MB is 100 MHz. The cores include local interrupt controller, local instruction and data memory and local instruction and data buses.

In a multi-core system we consider three types of components: processing cores (could be homogeneous or heterogeneous), private resources and shared resources (e.g. memories and interconnects). Processing cores and private resources have been profiled independently while for shared resources, specific bus monitors IPs have been implemented in order to identify the processing cores initiating the data transfers. The four MB cores cluster needs thus two shared DRAM bus monitors: one for instruction bus and the second one for data bus. These monitors allows us to dispatch the component energy to the core the current operation belongs.

We have considered the following representative benchmark applications to be executed on the reference design: a compression benchmark (COMPRESS), a CRC calculator (CRC), a FFT algorithm (FFT), a bit manipulation benchmark (BIT), a FIR filter (FIR), a Petri Net simulator (PETRI) and an auto generated code (AGC). These are self-contained software programs that do not require any additional libraries, operating or file systems. The self-contained benchmarks were needed to validate the energy interrupt, in order to avoid any interference with other programs, because the lack of an OS makes the results deterministic. While executing the benchmarks, performance and energy profiling have been performed.

The first test case consisted of FIR benchmark on multiple cores with four tests changing the location of instructions and data: (1) both benchmark's instructions and data are executed from DRAM, (2) both instructions and data are in BRAM, (3) instructions are executed from DRAM and data are stored in BRAM, and (4) instructions are executed from BRAM and data are stored in DRAM.

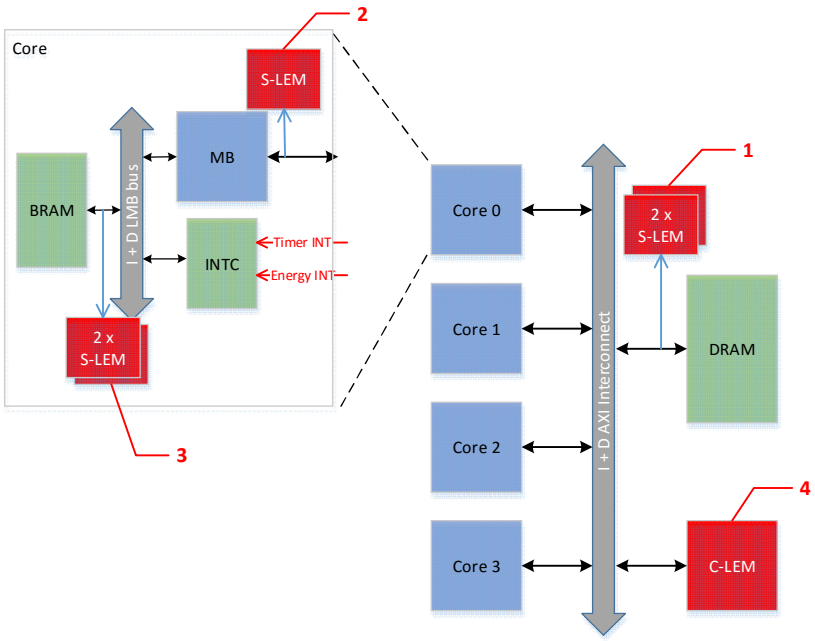

Fig. 100 Multi-core reference design

6.3 EXPERIMENTAL RESULTS

The experimental result of first tests executions are presented in Fig. 101 for FPGA core area usage and for frequency variation. As expected, a linear regression between dynamic power consumption of FPGA core and number of counter used (area or slices used) on one side and frequency on the other side. These results shows however that FPGA profiling can differentiate between power consumption of components at register level.

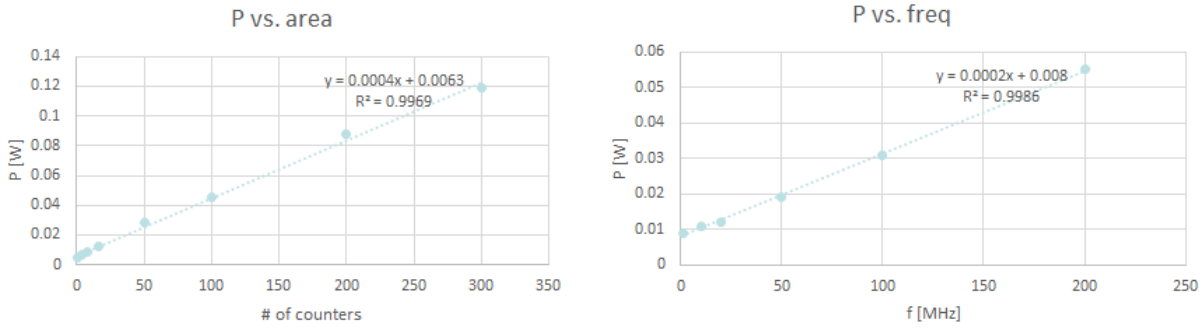

Fig. 101 (a left) FPGA power consumption vs. utilized Number of Slice LUTs; (b right) FPGA power consumption vs. frequency

In the third test bench we tried to see if there are differences in power consumption when are utilized logic blocks (Slice LUTs) or specialized DSP Slices. For this, we implemented an Nth-order Fixed-Point FIR filter. The implementation of the FIR filter is made so that it can use or not the DSP blocks. The implementation with DSP blocks uses existing built-in units for the following operations: addition, multiplication and signal delays.

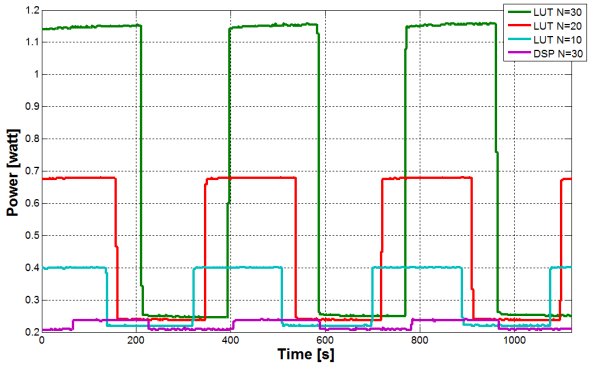

Fig. 102 FPGA power consumption for FIR Filter implementation using LUTs, N=[30, 20, 10] and using DSP, N=30

During filter testing procedure, 2 operating modes were used: filter on and filter off. The graphic representations of the FPGA consumed power can be seen in Fig. 102. We removed from the graphic, the configurations for N=[10, 20] implemented with DSP blocks, because the consumed power has a low value comparative with the rest of the values. We can conclude that using specialized blocks of the FPGA can decrease five times the consumed power. Also, the consumed power is proportional with the N (order of the filter).

Next, the obtained results for DDR shared component power and energy consumption are presented. Two test cases have been performed in order to emphasize power consumption and energy consumption of shared component: (1) execute memory transfers form Microblaze cores and (2) execute memory transfers from ARM cores. For each test case, several benchmarks have been performed: (1) sequential loads (LDR-seq), fixed ad-dress loads (LDR-fix), sequential stores (STR-seq), and fixed address stores (STR-fix). The memory transfers blocks sizes have been varied for each test. The average power consumption for load and store operations are shown in Fig. 103.

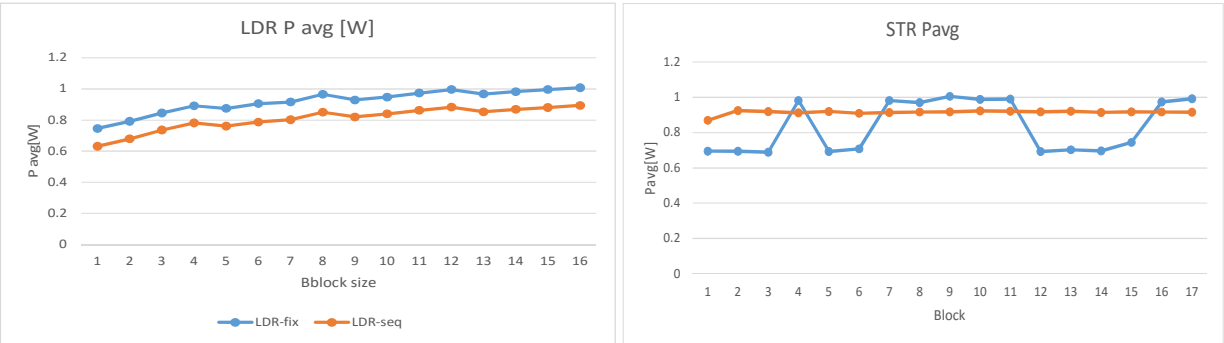

Fig. 103 DDR power consumption: (a left) LOAD transfers; (b right) STORE transfers

Measured data on Zynq 7020 development board for 2GB DDR3 SDRAM (MT41J256M8 – 32 Meg x 8 x 8 Banks) are identified through profiling:

- Power down: P_o = 45 mW
- Power idle (refresh): P_s = 585 mW

- Power of read (max): P_{rd} max = 1050 mW
- Power of read transactions (avg): P_{rd} avg = 0.010 x RD_usage_rate + P_s
- Energy of read transactions: E_{rd} = 11.596 pJ

A linear power consumption model for loads and stores can be applied based on measurements and a constant value for energy of load and stores operations has been obtained. The memory model of power consumption of load transfers function of memory usage is shown in Fig. 104. The energy consumption of load transfers function of loads transfer rate is shown in Fig. 105.

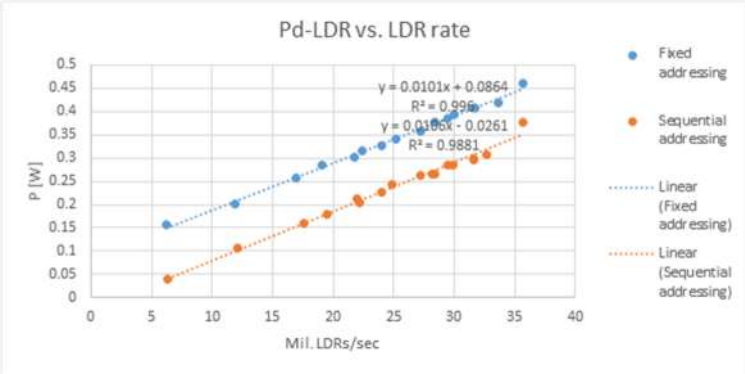

Fig. 104 Memory loads power consumption measurements

Similar results are obtained for store operations:

- Power of write transactions (max): P_{wr} = 1005mW
- Power of write transactions (avg): P_{wr} avg = 0.005 x WR_usage_rate + P_s
- Energy of write transactions: E_{wr} = 5.400 pJ

In conclusion we can estimate DRAM power by:

- PDRAM = C_1 x u_{RD} + C_2 x u_{WR} + Ps

and DRAM energy

- EDRAM = N_{RD} x E_{RD} + N_{WR} x E_{WR} + P_s x t

where u_{RD} is usage level of load transfers and u_{WR} is usage level of store transfers.

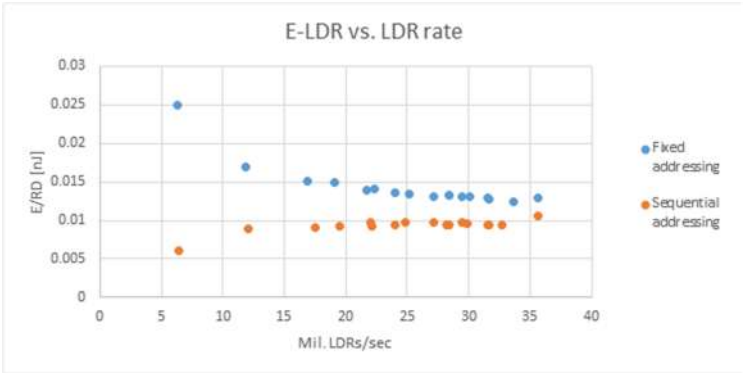

Fig. 105 Memory loads energy consumption estimations

Finally two RISC processors have been analyzed for power consumption and energy: PD_RISC (Synopsys Tools) and Microblaze (Xilinx).

The chosen instructions for testing the ALU unit of the processor are: **and16**, **or16**, **and**, **or**, **xor**, **add16**, **add**, **sub** and **mul**. Each of these instructions use the ALU unit of the processor in the own way, but there are many similarities between them. The measured power could vary, but only with minor variations. In time, there are absolutely no variations.

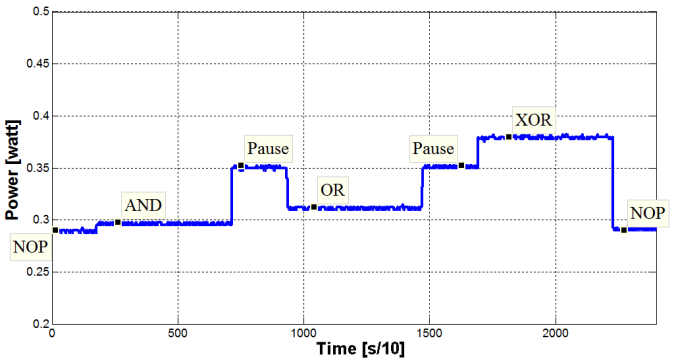

Fig. 106 Power consumption of a program sequence consists of following instructions: nop, and, or and xor

Fig. 106 shows the power variation for a testing program for following instructions: **and**, **or** and **xor**. For each of these instructions we have a large loop filled with only one instruction, followed by a pause loop. These pauses allow us to see the differences between the consumption of each instruction.

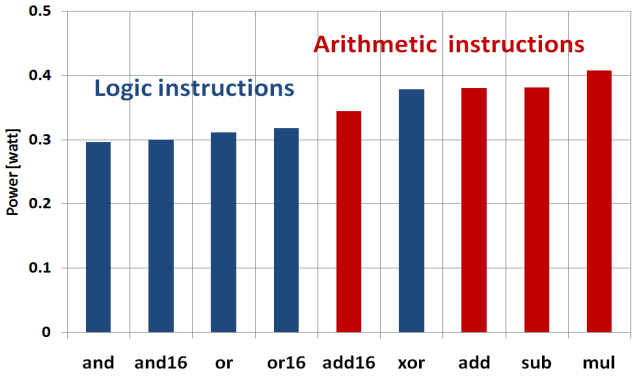

Fig. 107 Power consumption of ALU Instructions

The power consumption for ALU instructions can be seen in Fig. 107. The logic instructions have a lower consumption than the arithmetic instructions. The instruction with the most power consumption is the **mul** instruction, and if we consider only the logic instructions we have the **xor** instruction.

To access the memory, the PD_RISC processor has the following instructions: **lw**, **lh**, **lb**, **sw**, **sh** and **sb**. These instructions permit to store/load information on 32 bits, 16 bits and 8 bits. Consumption for all store instructions doesn't depend on the information stored in memory, but

for load instructions, the consumption could be different. Because of that, 3 types of experiments were made, which differ only by the loaded information.

In the first experiment, the information loaded from the memory is changed every time, so that each bit of the registry must switched from 1 to 0 or from 0 to 1 every time (e.g. suppose we load 8 bits of information, one time we load 10101010 , the next time we load 01010101 and so on). In the second experiment we loaded only 0 from memory, and in the third experiments we loaded only 1.

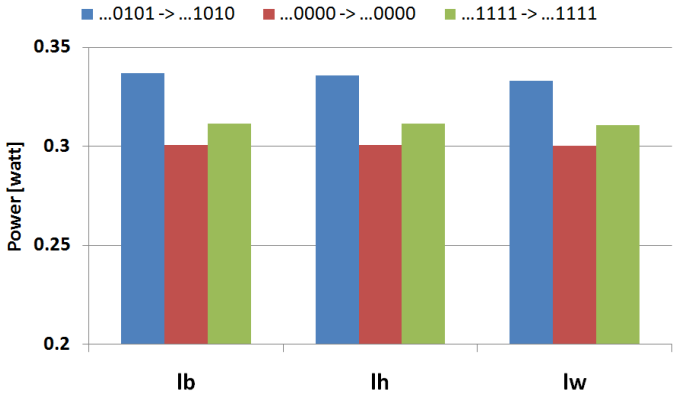

Fig. 108 Differences in power consumption of load instructions due to the loaded information. For 16 bits information we have: (blue) - one time load AAAA, the next time load 5555, and so on. (red) - load only 0000 words from memory. (green) - load only FFFF words from memory

The power consumption for all 3 experiments can be seen in Fig. 108. Regardless of the length of the information loaded from memory (8, 16 or 32 bits), the power consumption is higher when the bits of the register must switch when new information is loaded (first experiment).

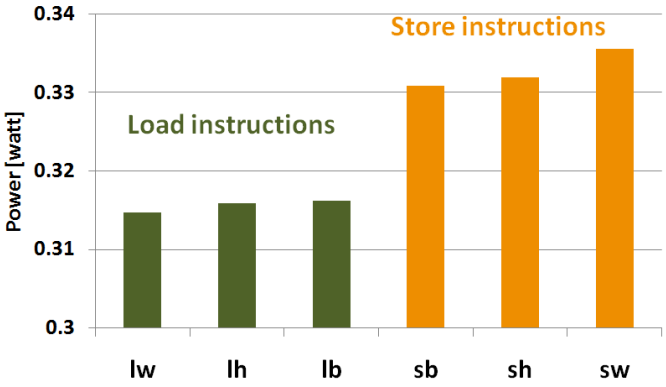

Fig. 109 Power consumption of memory access instructions

To establish a level of consumption for the load instructions, the obtained values for each instruction, in all of the 3 experiments, have been averaged. Putting together the store and the load instructions it can be easily seen that the level of power consumption for load instruction is lower that the level of power consumption for store instruction (Fig. 109).

In Fig. 109, different modes of load instructions show very close values of power consumption, but small differences can be observed. To investigate on this effect, standard cell based power simulation is performed to check power consumption of individual architectural components. Evaluation shows that the extra power values of **lb** and **lh** are consumed by the data masking logic in the MEM pipeline stage. Once the processor loads data word in 32 bits, the **lb** instruction masks 24 bits data out of 32 bits which consumes extra power. In contrast, the **lh** instruction masks only 16 bits data, which causes less overhead than **lb**. The **lw** instruction does not need any data masking, which consumes no additional power for data loading.

A number of 3 instructions existing in PD_RISC processor and used for programming control have been analyzed. These instructions are: **b**, **bcc** and **call**. The difference between **b** and **bcc** instructions is that the **bcc** instruction executes the branch based on a condition.

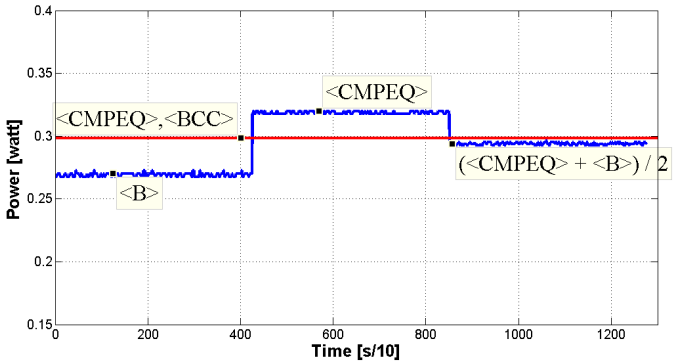

Fig. 110 Power consumption of cmpeq followed by bcc instruction is approximately equal with power consumption of cmpeq followed by b instruction

Internally, the **bcc** instruction means an analysis prior of the condition (resulted from another instruction), and after that it could make the jump. We wanted to verify this assumption, so it has been made the following experiment: we measure the power consumptions for **b** (unconditional jump), **bcc** instructions (conditional jump) and for a compare instruction **cmpeq**. The values for the consumption of **b** instruction and for the consumption of **cmpeq** instruction, together with the average of the two values, can be seen in Fig. 110 (the blue line). The red line is the consumption of one compare instruction followed by a **bcc** instruction. The red line is approximately equal with the end of blue line (the average value of **b** and **cmpeq**), the little difference represents the power consumption due to the prior analysis (which is part of a **bcc** instruction). This experiment validated the correctness of the measurements and also, gave us important details regarding how instructions are executed internally in the processor.

The last tested control instruction was the call instruction. Like **b**, it is an unconditional branch instruction, but, in addition, the call instruction stores the address of the following instruction into a register. Fig. 111 shows the consumption power values for all control instruction. As a reference, the power consumption for **nop** instruction was added.

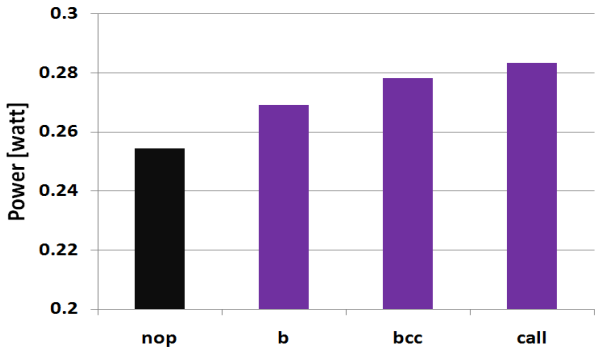

Fig. 111 Power consumption for control instructions

The last set of experiments was to determine the power consumption for compare and move instructions. Regarding compare instructions, it have been used instructions which compare register with an immediate signed/unsigned number and instructions which compare 2 registers. The chosen instructions were: **cmpltu**, **cmpeq** (register-register and register-number), **cmplt**, **srl** and **sll**.

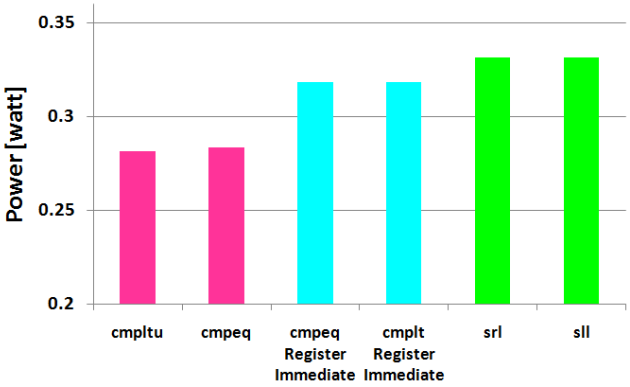

Fig. 112 Power consumption for compare and move instruction

The power consumption for all these instructions can be seen in Fig. 112. It can be observed three levels of power consumption: the lowest level is for compare instructions between 2 register, the medium level is for compare instructions between one register and one number, and the higher level is for move instruction.

The power consumption for all tested instructions can be seen in Fig. 113. The instruction with the most power consumption is **mul** instruction, and with the lowest power consumption is **nop** instruction. Another observation is that the instructions which are part of a group of instructions (e.g. arithmetic, logic, memory access, etc.) are grouped together in this chart.

For benchmarking purpose with FPGA power measurement, the test benches used on FPGA are simplified for gate-level simulation due to the long simulation time. Typically, each test bench contains one type of instruction with random operands, while the test bench is simulated for only 1,000 clock cycles. Compared to test benches used in FPGA measurement, such timing trade-off leads to inaccuracy in simulated power values, which is caused by the less switching activities in operand values.

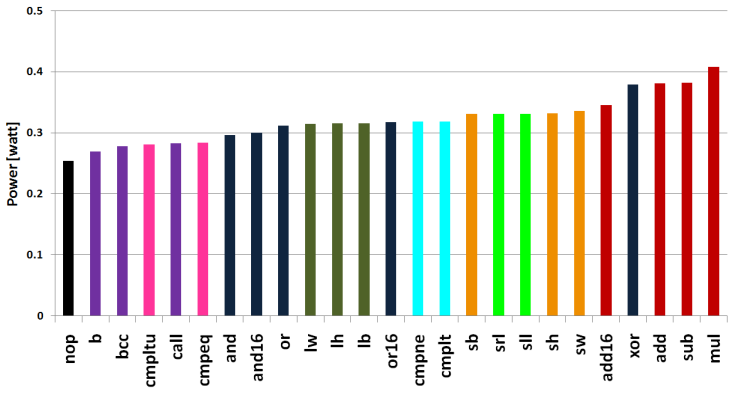

Fig. 113 Power consumption for all tested PD_RISC instructions

The correlation coefficient between physical measurements of a RISC core implementation on FPGA board (45nm) and simulated ASIC (45nm) power values, is 86.39%. Even though this value shows a strong correlation between the two series, the method based on physical measurements is recommended because of the reduced execution time and its higher accuracy and validation. The complete correlation matrix of the instruction level power consumption series achieved for different technologies and manufacturing processes is presented in Tab. 6. Very strong correlations between ASIC estimated series are observed, while good correlation between FPGA and ASIC exists too.

	130nm (faraday)	90nm (faraday)	65nm (faraday)	45nm (Nangate)	45nm (Atlys)
130nm (faraday)	1				
90nm (faraday)	92.11%	1			
65nm (faraday)	98.95%	92.53%	1		
45nm (Nangate)	98.67%	93.16%	98.37%	1	
45nm (Atlys)	85.78%	95.47%	84.34%	86.39%	1

Tab. 6 Power consumption correlation matrix

Finally, the power technology gap between the RISC core FPGA implementation and ASIC simulations for 45nm manufacturing process is between 200 to 1000 times larger.

Similar analysis has been carried out for the Microblaze core. Power consumption of main Microblaze instructions are shown in Fig. 114. Even MB core are optimized for FPGA designs, some differences in power consumption can be observed.

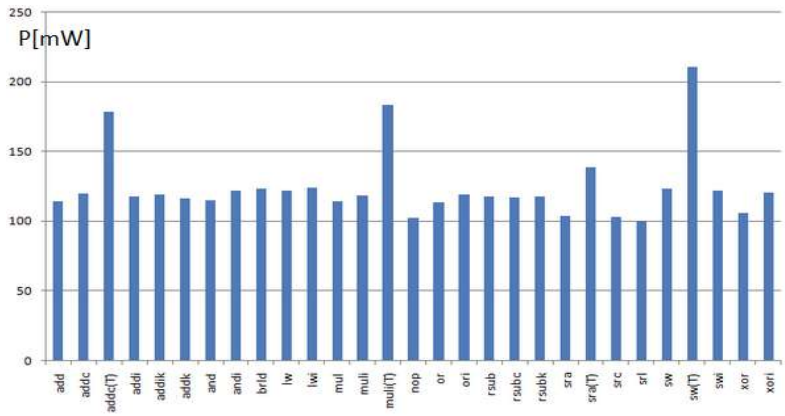

Fig. 114 Power consumption for all tested Microblaze instructions

6.4 RELATED WORK

At present, there are many techniques and tools used for monitoring an FPGA. Most of these tools have integrated power monitoring and energy consumption estimation solutions capable of doing most of the work. In [Tong2008], a survey of profiling tools is presented and they are categorized as follows: software-based, hardware-based and FPGA-based. The software-based profiling is widely used in order to analyze the performance of application code and can be further classified in two approaches: simulation and insertion of instrumentation code. The hardware-based profiling tools [Tong2008] make use of hardware counters available on-chip that are used to monitor certain events that occur while an application is executed. The events that can be checked are: pipeline stalls, cache misses, memory accesses and others. FPGA-based profiling tools [Tong2008] make use of the soft-processor available on FPGAs, which is used as target processor for the execution of software programs. The profiling process is as follows: the software is executed on the target soft-processor by the designer and after that, the information about performance is captured using the hardware for profiling available on-chip.

Many of these tools used a soft-processor in order to perform the evaluation for a given design. The estimation is done based on code analysis and other external instruments. LEAP is presented as an FPGA-based hardware profiler being developed as a non-intrusive and area-efficient tool [Aldham2011]. In order to analyze an executed program, LEAP monitors the instruction bus of the processor together with the program counter. Its objective is to find a perfect correlation between the routines and the counters, based on the starting address for each function.

In [Genser2009], the authors developed two profiles for each software application: a power estimation-based profile and a power measurement-based profile. The estimation-based profile is less accurate because it's based on power modeling, but it provides higher flexibility. The other profile, provides extremely accurate results, but requires additional equipment.

A framework made by many abstraction layers is presented in [Powell2013]. This framework is useful for the design space exploration process. Inside the framework many units are presented,

like: control unit, instruction management unit, data processing unit or memory management unit. Each unit has different metrics and builds a separate power profile.

Another energy estimation method is via instruction level energy profiling [Laopoulos2003, Kavvadias2004]. There, every instructions' power cost is broken into multiple costs (e.g. base cost, interaction cost, operand costs, etc.). This approach tried to capture all the dependencies between instructions. However, this method is difficult to implement because of the multiple dependencies between instructions (e.g. data dependencies, processing features, cache type, multiplier design, etc.)

In [Ou2004] the authors propose an instruction level profiling methodology on FPGA designs. The proposed methodology can rapidly estimate the energy dissipation of software programs running on FPGA implemented soft processors. In our work we address energy profiling of multi-core architectures. In [Afifi2014], a method is proposed to estimate the dynamic power consumption of Microblaze soft core used in embedded designs for FPGA. The proposed method is based on experimental test bench of implemented designs where the processing unit is repeated distinctly, and on real power measurements are captures.

The authors of [Pallares2012] focus on instrumentation and measurement of FPGA designs for power quality measurements. In [Atitallah2013] the authors present a global framework for power and energy estimation and optimization of heterogeneous multiprocessor system-on-chip. Within this framework, a power modeling methodology is defined, and an open platform is developed. The proposed methodology takes into account all the embedded system layers: the software, the hardware, and the operating system.

Our approach is based on physical power measurement of the FPGA device. A similar approach was used in [Nakutis2009], for several types of embedded systems. However, our method is not restricted on designs which use soft processors and can be performed on any FPGA device. Similar work has been performed on Zynq development board for power control and monitoring [Beldachi2014]. Power monitoring and estimation do not target only processing cores but also memory and interconnect components [Schurmans2013].

6.5 SUPPORTING GRANTS, RESEARCH TEAM AND SCIENTIFIC OUTCOMES

Project title: GreenEr Mobile Systems by Cross LAyer Integrated energy Management

National competition: CHIST-ERA

Implementation period: 2012-2015

Project acronym: GEMSCLAIM

Project code: CHIST-ERA/1/01.10.2012

Budget: 250 000 EUR

Project summary: The GEMSCLAIM project aims at introducing novel approaches for reducing the "greed for energy" of modern battery powered systems, thereby improving the user experience and enabling new opportunities for mobile computing. Mobile terminals and consumer devices are among the fastest growing markets in computing. In the long term, further growth is endangered by the "power/ energy wall". The purpose of GEMSCLAIM is to explore new techniques in energy optimization via an interdisciplinary vertical approach: a novel combined optimization across the major HW/SW system layers (compiler/OS/HW platform).

The ever-growing need for energy efficient computation requires adequate support for energy-aware thread scheduling that offers insight into a systems behavior for improved application energy/performance optimizations. Runtime accurate monitoring of energy consumed by every component of a multi-core embedded system is an important feature to be considered for future designs. Although, important steps have been made in this direction, the problem of distributing energy consumption among threads executed on different cores for shared components remains an ongoing struggle. We aim at designing a generic low-cost and energy efficient hardware infrastructure which supports thread level energy consumption monitoring of hardware components in a multi-core system. The proposed infrastructure provides upper layers (operating system and application threads) with per thread and per component energy accounting API (Application Programming Interface), similar with performance profiling functions. Implementation results indicate that the proposed LEM (Load and Energy Monitor) adds around 10% resource overhead to the monitored system. Regarding the power estimates, the one derived by LEM achieve a correlation degree of more than 95% with the ones obtained from physical power measurements.

The research team involved in the project included experienced researchers, young researchers, external researchers and students:

PhD. Marius MARCU (local partner director)
PhD. Oana AMARICAI-BONCALO
PhD. Alexandru AMARICAI-BONCALO
PhD. Cosmin CERNAZANU-GLAVAN
PhD. Sebastian FUICU
PhD. Razvan BOGDAN
PhD. Gabriel GIRBAN
Eng. Lucian BARA
Eng. Madalin GHENEA
Eng. Marian IONASCU

Partners:

Innsbruck University (Lead partner)
Queen's University Belfast
RWTH Aachen University

The contributions of this work are as follows:

- hardware infrastructure for dynamic energy consumption monitoring in a heterogeneous multi-core system with per-thread energy accounting;
- energy interrupt specification and design;
- a use case on the software side (OS and drivers) for run-time per-thread energy accounting implementation on FPGA; and
- validation of proposed infrastructure on a high-end FPGA board with physical energy measurements.

This section is based on the following papers published by the author: [ISI5], [BDI2] and it has been submitted to ISCAS 2016 conference.

6.6 CONCLUSIONS AND FUTURE WORK

FPGA development boards are broadly used nowadays to implement target platforms prototypes and test them on real hardware. In our work we investigated the possibility to profile energy of target hardware designs using FPGA synthesis and we presented energy profiling results for a FPGA development board. The main goal of our study was to investigate the power measurement and profiling capability of FPGA development boards.

The processor to be profiles has been implemented on FPGA and different benchmarks programs have been executed. The goal of these measurements has been to investigate the correlation between dynamic power consumption measurements of RISC processor instruction set on FPGA board with the ones estimated for the same processor implemented in a different technology.

The processor instruction set is divided into several groups: arithmetic-logic instructions, memory access instructions, control instructions, compare and move instructions. An interesting observation is that the instructions from a group of instructions have similar power consumption. If we chart the power consumption for all the instructions we can observe that each group of instruction has a certain level for power consumption (distinct from other groups). The instruction with the highest power consumption is mul instruction, and with the lowest power consumption is nop instruction.

The correlation coefficient between the FPGA 45nm power measurements and ASIC 45nm power estimation is 86.39%. The benefits of this work are two folds. On one hand we can estimate dynamic power consumption by physical measurements, while on the other hand, we can validate power consumption models using real measurements.

As a future work, it is important to determine how the power estimation accuracy scales with increasing program complexity. Furthermore, an accurate power monitoring and management solution based on fine hardware parameters tuning will be implemented.

7 LOW-COST HARDWARE INFRASTRUCTURE FOR RUNTIME THREAD LEVEL ENERGY ACCOUNTING

7.1 OVERALL DESCRIPTION AND RESEARCH OBJECTIVES

The ever-growing need for energy efficient computation requires adequate support for energy-aware thread scheduling that offers insight into a systems behavior for improved application energy/performance optimizations. Runtime accurate monitoring of energy consumed by every component of a multi-core embedded system is an important feature to be considered for future designs. Although, important steps have been made in this direction, the problem of distributing energy consumption among threads executed on different cores for shared components remains an ongoing struggle. We aim at designing a generic low-cost and energy efficient hardware infrastructure which supports thread level energy consumption monitoring of hardware components in a multi-core system. The proposed infrastructure provides upper layers (operating system and application threads) with per thread and per component energy accounting API (Application Programming Interface), similar with performance profiling functions. Implementation results indicate that the proposed LEM (Load and Energy Monitor) adds around 10% resource overhead to the monitored system. Regarding the power estimates, the one derived by LEM achieve a correlation degree of more than 95% with the ones obtained from physical power measurements.

7.2 PER-THREAD ENERGY ACCOUNTING

7.2.1 Hardware infrastructure

The proposed PTEA solution, called Load and Energy Monitor (LEM), implements a distributed sensor network through a dedicated bus interconnect based on the open hardware Wishbone Bus (WB) specification [OpenCores2010], which is both cost-effective and light-weight. It consists of a generic hardware monitoring infrastructure which is able to connect the distributed sensors attached to the components of an embedded target design. LEM monitoring infrastructure allows software layers to access at runtime, both the component level usage and the energy estimates. LEM control and configuration is centralized, thus providing a unique interface to the upper layers.

The LEM infrastructure building blocks are: LEM sensors (S-LEM), LEM controller (C-LEM) and LEM interconnect (LEM Bus) (Fig. 115). The sensors are hardware components that collect data from the underlying system at a programmable high-speed rate. Furthermore, they perform component level performance or energy accounting computations; thus, allowing fine grain component power tuning. The sensors sampled values may be switching activity, performance counters, and power or temperature physical measures. Therefore, the proposed sensors offers high flexibility to system designers. For the considered uses cases in our work, we have implemented only energy accounting sensors, because we aimed at validating per-core energy accounting as

hardware support for PTEA. Sensors can be attached to individual cores, accelerators, or shared component components of the target system such as memories or buses.

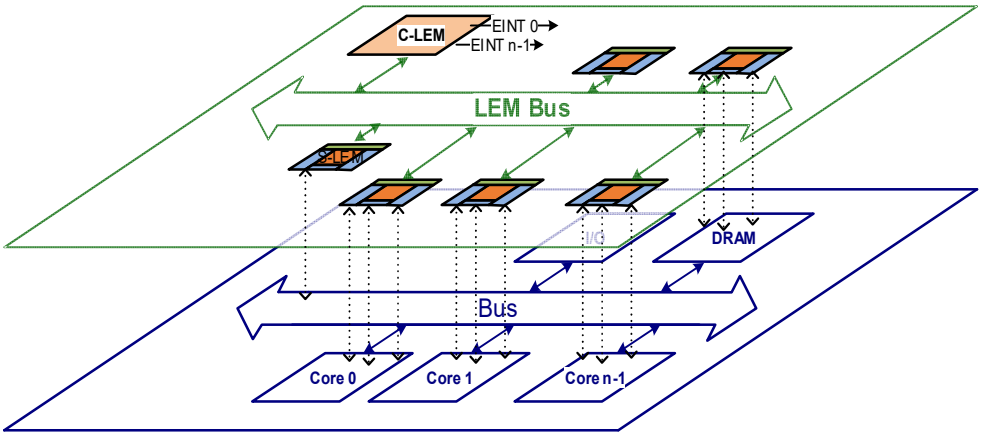

Fig. 115 Overall energy monitoring architecture

C-LEM performs energy accounting at core level, collecting and summing energy consumption values from both local sensors, attached to cores' local buses and memories, and shared sensors, attached to shared components, such as DRAM memory or system busses.

Two important features of LEM infrastructure are non-intrusiveness and energy interrupts. Non-intrusiveness is achieved by weak coupling between the monitoring infrastructure and the target system (Fig. 100), due to the special sensor design. Energy interrupts are generated for each core in the target design when the energy budget corresponding to the core has been consumed.

The sensors implement a standard interface for the various performance and/or consumption monitoring support existing in hardware. The sensors are composed of (Fig. 116.a): (1) sensor interface used to connect the sensor core with the central LEM; (2) sensor back-end (with some processing and tuning support - hardcoded coefficients and/or look-up tables) with limited energy consumption accounting and (3) hardware interface probes which is component or bus specific. The sensors can be configured to provide energy, power or raw data. Energy accounting is performed by summing up the samples from the component until then next poll from the master LEM controller. Power monitoring is performed by averaging the samples from the component until the next read from the master controller. Raw data from low-level component monitors are passed directly as instantaneously values to the LEM controller.

Sensor back-end either reads the energy consumed by a component (some processing cores offer this information through some dedicated ports), or derives the energy consumed by adding for each accounted event a value from a stored table with a priori computed values. Some approaches which rely on software post-processing of C-LEM data, similar in functionality as the ones used in HEMA tool [Choi2012], can be used to reduce the resource overhead for the monitoring infrastructure.

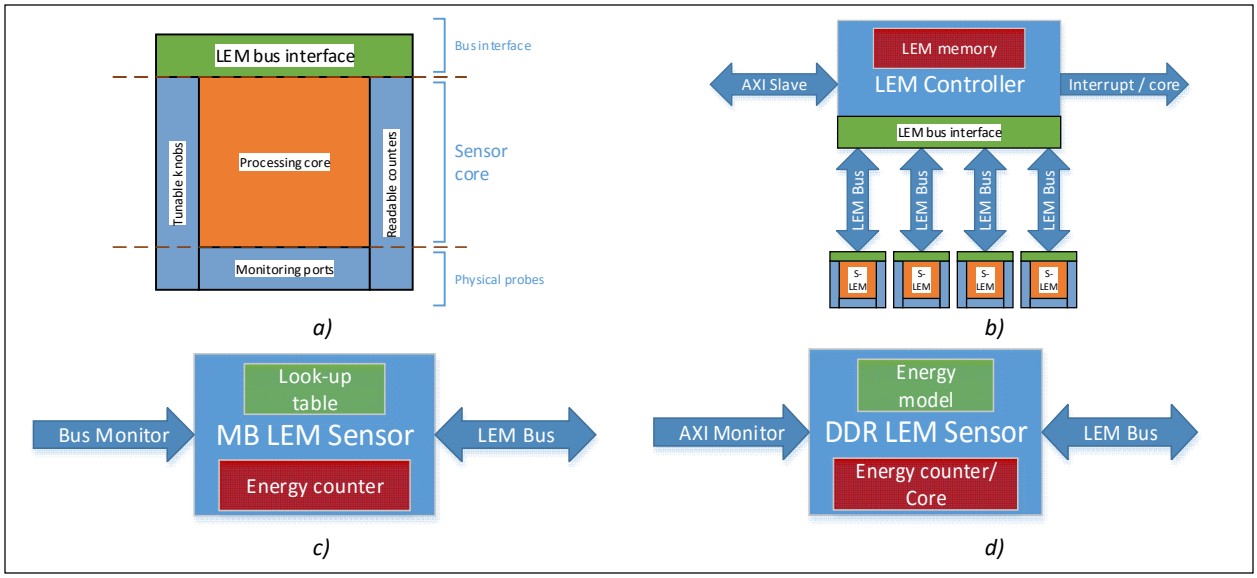

Fig. 116 LEM components: a) Sensors' architecture; b) C-LEM; c) MB instruction sensor; d) DRAM shared data sensor

Depending on the power metering infrastructure available on the board the LEM sensors can be attached to physical sensors or can implement energy models. Our solution aims at deriving energy consumption by correlating event accounting (i.e. event counters at various levels in our system) with component based energy in the pre-computed look-up-tables. Using monitoring ports, LEM sensors can access underlying monitoring components (e.g. performance counters, physical sensors, or events monitors).

C-LEM (Fig. 116.b) controls and collects sensor data from existing LEM sensors and provides the OS with structured access to the sensors' data. The OS communicates with the central LEM unit through the global memory address space for sensor control and sampling. Data received from S-LEM is aggregated into the C-LEM local memory. Each component of the target system has a reserved 128 bits memory structure, including timestamp, component usage, processing energy, and data movement energy. One or more sensors collect measurements for every component, while C-LEM aggregates these values at the component level. For each component two energy values are accounted: component operation energy and data moving energy. These values are collected from the sensors monitoring the respective component and its interfaces. Each memory location stores the last value calculated from all the sensors that are attached to the core itself or to a shared component accessible to the core.

Furthermore, C-LEM is also highly parameterized, and it can be programmed by the underlying software same as the router from [Madduri2009]. Dedicated registers allow the programming of the LEM for: enabling/disabling monitoring for a component, enabling/ disabling energy interrupt, changing the sampling rate, controlling the power model or selecting the sensors accounting mode.

LEM bus is based on WB standard [OpenCores2010], having one master (C-LEM) and several slaves (S-LEM, up to 256 in the current implementation). It has a configurable number of data lines for read and write channels, which provides C-LEM with efficient access to S_LEMs. As

read operations are more frequent, we have selected an increased bus width for this channel (32 in the current implementation), with respect to the write channel (8 bits). Fig. 116.b highlights the C-LEM and its LEMbus master and AXI slave interfaces. C-LEM has also an AXI slave interface, thus, providing access for the target system to read data and write commands for the LEM. The cores can poll C-LEM for their current load and energy counters through the AXI interface.

There are currently three classes of sensors used by LEM: (1) instruction sensors (core processing sensors), that monitor an instruction bus and calculate the energy of each instruction executed by a single processing core; (2) shared data sensors, that monitor a shared data bus and calculate the energy consumption of data transfers for each processor monitored; (3) local data sensors, that monitor a data bus that corresponds to a single processor. The LEM reads data from each sensor sequentially and aggregates the data depending on the type of the sensor and stores it in the energy counters of C-LEM memory.

Instruction sensors provide LEM with support to measure or estimate the processing energy of the cores they are attached to. These sensors are specific to the processing core they monitor and the instruction bus they are connected to. However, being modular, S-LEM sub-components have a high degree of flexibility: the S-LEM processing core can be connected to different busses only by changing the bus interface monitor. Instruction sensors included in the FPGA design target the Microblaze (MB) processor, a soft-core provided by Xilinx [Xilinx2014]. Microblaze's S-LEM is implemented as a virtual sensor that monitors the core's instruction bus and partially decodes the MB opcodes in order to associate estimates with the execution energy (Fig. 116.c). Energy counter is incremented with the energy amount associated to the instructions executed by the processor. Energy values stored in the S-LEM look-up table have been computed through energy profiling on the target FPGA board.

Shared data sensors attached to the shared components of the target system allow per-core energy estimation of these components. These sensors are able to identify and dispatch the amount of energy consumed by the shared component to the core that is responsible for the energy consumed by the component (Fig. 116.d). Shared data sensors implement internally one energy counter per core, whose contents are summed up at the core level, within the C-LEM.

Each data transfer and instruction execution made by a processor have an energy budget. The aforementioned data transfers and instructions are monitored by several sensors which transmit the energy consumed to C-LEM. Component load and energy counters can be accessed directly by cores through their memory space or an interrupt is generated when a specific energy threshold has been reached. Energy interrupts are hardware interrupts that become active when the energy budget allocated to a specific component or core has been consumed.

LEM can be programmed to send an energy interrupt to a processor which is being monitored when an energy budget threshold is reached. The threshold value is stored by one energy quantum register allocated per core. The energy budget value is compared to the current energy consumption, which is calculated by summing the values stored in the Instruction Energy Register and the Data Energy Register. If the current energy consumption exceeds the energy budget value and the LEM interrupt option is active, then an interrupt signal is generated for one clock cycle

and is captured by an interrupt controller, this is transmitted to the corresponding processor. After the deactivation of the interrupt signal, the LEM energy registers corresponding to the current processor are reset. The LEM can monitor several processors simultaneously and can generate independent interrupts for each processor when it reaches their corresponding energy threshold. Each energy interrupt can have a different interrupt handler; thus, a higher degree of flexibility is achieved.

7.2.2 Implementation use-case

LEM infrastructure is a hardware solution that can be customized for different applications. It has been further customized and used to implement PTEA in a multi-core system. In a multi-core system we consider three types of components: processing cores (could be homogeneous or heterogeneous), private resources and shared resources (e.g. memories and interconnects). In a multi-core hardware environment, every core has a unique core ID. Our idea is to use this ID in hardware transaction with shared components to identify the processing core which will be charged with the energy budget of the current transaction. The LEM sensors connected to these shared components will use master ID to account per core energy consumption.

Modern interconnects like Wishbone (open source) and AXI (ARM) allow system designers to attach meta-information to each transfer. For example, Wishbone bus standard specifies user-defined tags to apply extra information to each bus cycle [OpenCores2010]. On the other hand, AMBA open specification [ARM2003] associates implicitly hardware IDs of the master to each bus cycle. ARID, RID, AWID, WID and BID bus signals are carrying ID tags of the read address, read, write address, write and response bus transfers. Hence, it makes sense for the LEM sensors of shared resources to use the master ID of the access to split the energy consumed for shared components.

Using the LEM infrastructure and the ID tagging bus support, PTEA implementation can be split in two steps: (1) Per core energy accounting of processing cores and shared resources based on hardware support; (2) Per thread energy accounting implemented at OS level during context switching, using the provided LEM drivers. LEM driver provides OS with an API similar to existing performance profiling counters functions. The LEM driver interface is based on start/stop accounting operations.

When a thread context switch occurs, the OS will store the energy counters of the current thread and will restart the counters for the next thread. While a thread is executed by a core, per core energy accounting implemented in hardware will be used to account for the energy of a running thread. Considering that a core will execute only one thread at the time, the coordination between OS and HW will account for the thread level energy in a multi-core/multi-threading execution environment.

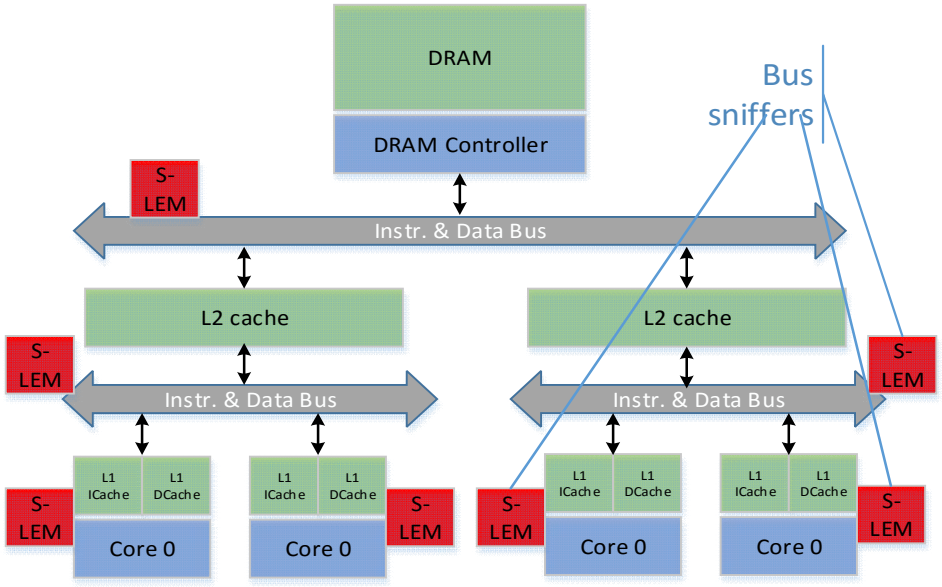

Fig. 117 Memory hierarchy monitoring

LEM hardware infrastructure comprises of bus/interconnect sniffers (S-LEM). These monitor for the number and type of accesses for each shared components. Energy accounting is estimated using look-up-tables which correlated the energy consumed with the number and type of accesses. We further discuss the PTEA use case for the example target platform used in [Lui2013] – Fig. 117. In case core0, from cluster0, has a cache miss in L2 - cache, three events are reported by three sensors: access L1 - cache (by the processing core's sensor), access L2 - cache (event reported by the cluster bus: L2 cache access), access memory controller (event reported by the system bus). Each of these events are accounted on behalf of core0. The proposed approach differs from the one described in [Lui2013] by the fact that events such as dirty line, cache line eviction, cache hit or miss are not monitored solely on the shared component side. These are a direct consequence of our proposed accounting method, and are transparent for the infrastructure. For example, a cache line eviction require a number of events on the system bus; the shared cache sensor for the processing core (identified by master ID) is notified by the sensor on the system bus that energy needs to be accounted on its behalf. This mechanism is hierarchical. Furthermore, the means for these types of notifications represent transactions on the LEM infrastructure. Therefore, the bus topology of the LEM infrastructure is similar with the one in the monitored system.

So far we discuss the possibility of using LEM as a passive monitoring component, much like the way performance monitoring counters are used by system software. In brief, at context switch dictated by scheduling time intervals, the OS kernel on each core queries C-LEM to retrieve energy sensor readings and attributes energy consumption to running threads. While this approach can accurately account energy consumption at the thread level, it provides limited support to implement energy budgeting policies which are crucial in optimizing energy efficiency of embedded platforms [Zeng2002][Snowdon2009]. This is because the OS can account of energy consumption only by sampling at time-based scheduling intervals. During such an interval, the OS has no control over how much energy a thread spends, thus it cannot enforce energy budget limits.

We extend LEM to implement what we refer to as the energy interrupt. In the same way a programmable timer generates an interrupt whenever its time slice depletes, LEM is extended with an energy slice and generates an energy interrupt when this energy slice depletes. Specifically, LEM is extended with an extra register per core (associated with a unique core ID) to store the energy slice value. Recall core IDs uniquely identify thread IDs in the OS, since only one thread can be running on a core at a time. The OS sets LEM's energy slice register to the available budget for this core which corresponds to the allotted energy budget for the core's executing thread. LEM decreases the energy slice by monitoring the energy consumption due to the core's activity across all hardware components. When the energy slice is depleted, LEM generates an energy interrupt to the core, to be handled by the OS. The OS interrupt handler reads the LEM's sensor values per hardware component for a breakdown on thread's energy consumption and overwrites the LEM's energy slice register with the new energy budget value for the scheduled-in thread, according to the energy budgeting policy. A further extension, left as future work, is to enable energy slices for each hardware component instead of accumulating energy consumption across all of them. This will permit even finer control on distributing the energy budget among hardware components, to be leveraged in component-level energy budgeting policies.

7.3 EXPERIMENTAL RESULTS

The LEM infrastructure has been implemented on the Xilinx ZC702 evaluation kit, with Xilinx Zynq-7020 device. Xilinx Vivado 2014.4 toolset has been used for the implementation of the system. The development board has built-in power monitoring sensors for the power lines of the main components: processing system cores (ARM cores), programmable logic core (FPGA fabric), DRAM, I/O etc. Four target systems similar with the reference design (see Section 5.2) have been analyzed in order to estimate the area/performance/energy overhead of the LEM implementation compared with monitoring free situation. Furthermore, LEM implementation was compared with other solutions that can provide support for energy accounting implementation: existing Microblaze (MB) performance counters and Xilinx IP AXI PerfMon. The four systems feature 1, 2, 4, and 8 cores respectively, each being extended with all the infrastructure needed to estimate energy consumption per core.

The implementation overhead of the Microblaze SoC system featuring 1,2,4, and 8 cores respectively is presented in Fig. 118. Fig. 118.a presents the slice cost for the system with the following monitoring configurations: no monitoring, MB performance counters monitoring, Xillinx bus performance monitors, and the proposed monitoring infrastructure. The cost for the 8 core system with performance monitors did not fit the Zynq board resources. Compared to other monitoring solutions LEM infrastructure gives a better area/power consumption tradeoff (Fig. 118.b). With respect to the Xilinx bus perfomance monitors, it presents up to 30.7% less slice based resources, while offering the same degree of observability. Compared to the MB performance counters, it present a 11.43 % increased cost. However, the proposed infrastructures has the following advantages: (i) it has increased observability, because it provides performance information for other components in the system (bus, memory controller, etc), (ii) it has better performance, as it requires less instruction cycles for configuration and reading.

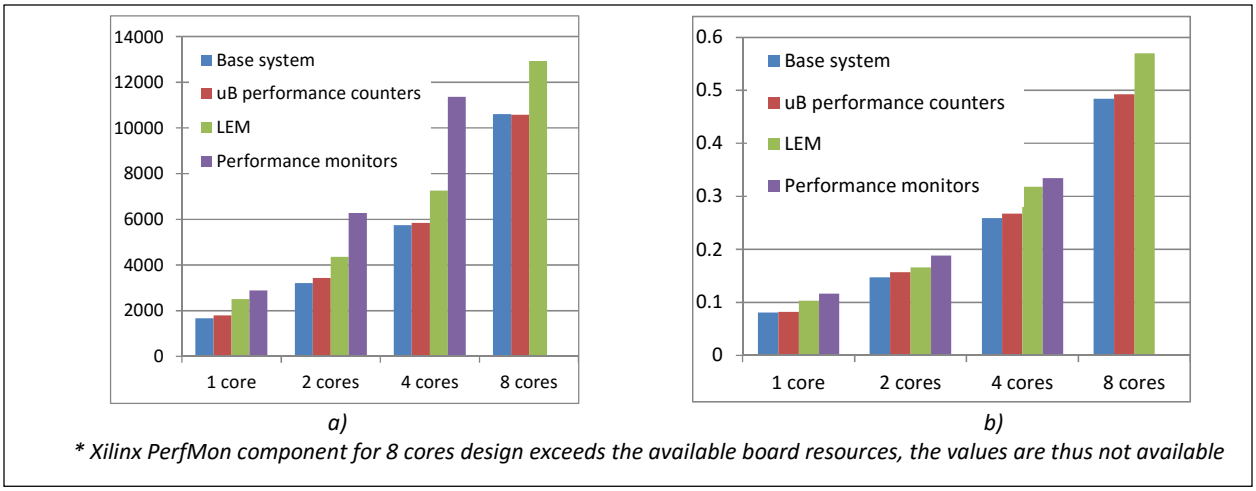

Fig. 118 Cost estimates for the system with different monitoring infrastructures: a) Number of used slices; b) Static power consumption

The reference system design for PTEA validation is presented in Fig. 100 and has four Microblaze soft cores. The clock frequency for the MB is 50 MHz. The cores include local interrupt controller, local instruction and data memory and local instruction and data buses. Local interrupt controllers serve two interrupt sources: time interrupt (1 ms) and energy interrupt (~100 nJ).

The reference design implements the LEM infrastructure attached to the 4-cores design. The system with four MB cores need two shared DRAM sensors Fig. 100-1 (one for instruction bus and the second one for data bus) and three more local sensors per core (one DRAM instruction sensor - 2, one BRAM data sensor and one BRAM instruction and data sensor - 3). In total LEM implements 14 sensors and one C-LEM (4) with 16 ports.

To validate the LEM infrastructure, standalone benchmarks from the WCET project [WCET] have been selected. We have considered the following representative benchmark applications: a compression benchmark (COMPRESS), a CRC calculator (CRC), a FFT algorithm (FFT), a bit manipulation benchmark (BIT), a FIR filter (FIR), a Petri Net simulator (PETRI) and an auto generated code (AGC). These are self-contained software programs that do not require any additional libraries, operating or file systems. The self-contained benchmarks were needed to validate the energy interrupt concept, in order to avoid any interference with other programs, because the lack of an operating system makes the results deterministic.

The benchmarks have been run two times on two distinct Zynq boards. Each benchmark has been executed on the reference system on one core, two cores and four cores. Regarding the storage of data and instructions for each benchmark, the following cases have been considered: (1) both instructions and data are located in DRAM, (2) instructions are executed from DRAM and data are located in BRAM, (3) instructions are executed from BRAM and data are located in DRAM and (4) both instructions and data are located in BRAM. BRAM is the local memory of each core, having the same performance as an L1 cache. L1 and L2 cache levels have been disabled during the tests in order to observe the energy impact of shared/local components. While executing the benchmarks, performance and energy profiling have been performed. For each test set timer ticks,

LEM energy quants and physical measurements have been recorded for further analysis. However, physical measurements needed many sequential runs of the benchmarks with the same parameters in order to let the system stabilize. In total more than 48 tests have been performed for each benchmark, each test lasting 1-2 minutes. LEM sensors have been calibrated using existing energy profiling techniques as described in the previous section.

The first test case consisted of FIR benchmark on multiple cores with four tests changing the location of instructions and data: (1) both benchmark's instructions and data are executed from DRAM, (2) both instructions and data are in BRAM, (3) instructions are executed from DRAM and data are stored in BRAM, and (4) instructions are executed from BRAM and data are stored in DRAM. We aimed at determining the correlations between the power estimation provide by LEM infrastructure with the physical measurements on the Zynq board. The graphic in Fig. 119.a depicts the correlation obtained for the FIR benchmark. Overall correlation value for this benchmark is 96.57%. For this case, LEM sensors have implemented simple MB, DRAM and BRAM energy models based on look-up tables and linear regression, respectively. However, this result validates the LEM infrastructure both as (1) a mean for new energy models calibration and evaluation based on runtime hardware measurements and (2) online system hardware profiling based on physical onboard sensors or virtual model based sensors.

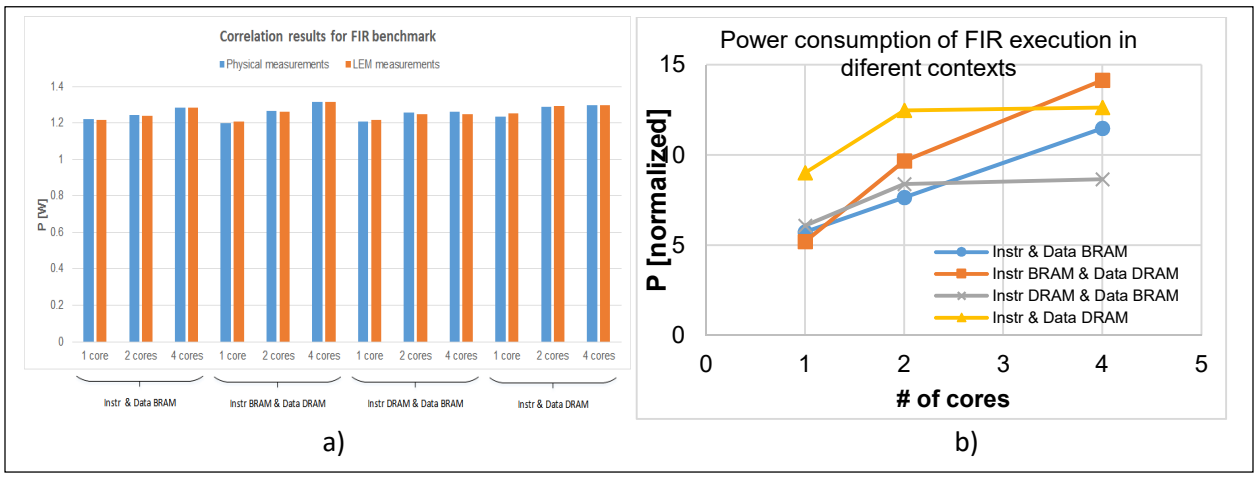

Fig. 119 Power consumption measurements for FIR benchmark: a) Correlation results b) LEM normalized power

Fig. 119.b depicts an analysis for the online system hardware profiling for systems with 1 MB core, 2 MB cores and 4 MB cores which execute the FIR benchmark. When instructions are executed from local BRAM memories, the performance and the energy consumption of the system scales linearly with the number of cores. This remark shows the efficiency of L1 instruction caches (which for FPGA systems are implemented in BRAM); this result also validates the proposed monitoring environment. On the other hand, when instructions are executed from DRAM, the system is limited both in performance and power consumption at two cores.

The second test case aimed at emphasizing the energy efficiency of the benchmarks' executions in different system contexts: number of cores, location of instructions and data. In order to characterize benchmarks' executions in different contexts we accounted for runtime timer ticks

and energy quanta (number of energy interrupts compared with number of timer interrupts) needed to finalize the benchmarks. In Fig. 120, FIR benchmark performance and energy characterization is presented. Energy consumption and performance of the benchmark significantly depends on the data and instructions location. The system consumes more energy when executes instructions from DRAM memory. In this context, two cores shows the best energy efficiency, while 4 cores increase the penalty in energy and performance due to the DRAM bottleneck. Moving instructions from DRAM to BRAM increases the energy efficiency by a factor of 4.

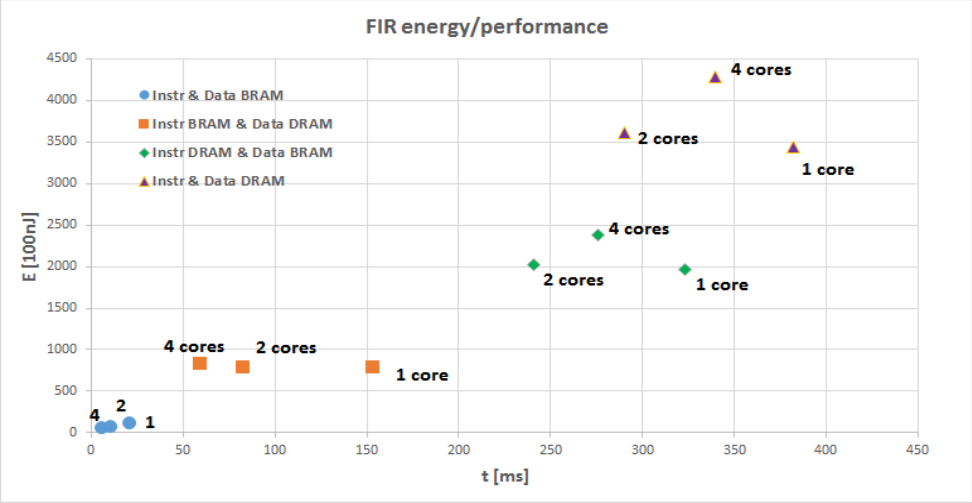

Fig. 120 Energy efficiency results for FIR benchmark

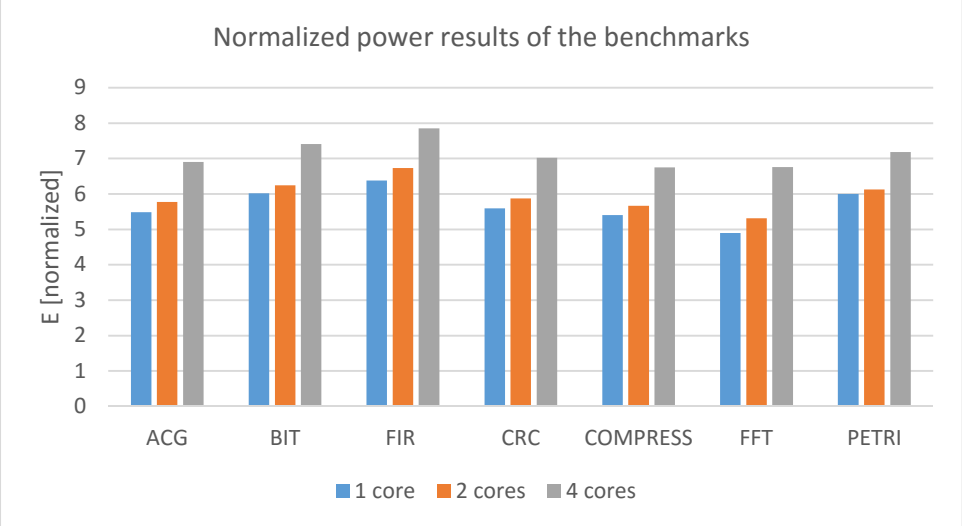

Fig. 121 Energy profiling of WCET benchmarks

The third test case aimed at validating the LEM infrastructure ability to do software level profiling. Fig. 121 shows normalized energy consumption of all WCET benchmarks executed from DRAM. Normalized energy is used in order to be able to compare different benchmarks. Differences between energy consumption results of benchmarks executions can be observed, which shows LEM infrastructure capability to profile software.

7.4 RELATED WORK

Hardware monitoring infrastructures have been proposed in the past, with results reported for both FPGAs and ASICs. These solutions have to provide several main features: power and performance meters (physical or model based sensors), data collection (interconnects), control unit (configuration and processing) and finally, software drivers (API).

The work presented in [Bouajila2012] uses a customized token ring interconnect for monitoring and actuation. Two lines are dedicated for communication control (token and valid), with as many as desired data lines possible (based on the transmission speed and/or resources committed for the infrastructure). This solution is simple, with a limitation for the transmission delay: it increases linearly with the number of nodes. Research in [Madduri2009][Zhao2011], relies on a network on a chip (NoC). Hardcoded routing tables and two types of monitors (i.e. data pull and data push) try to limit the overhead introduced by the NoC. Another approach [Ituero2012], proposes a light-weight monitoring system relying on a single-wire interconnect network, where monitoring components take turns to send data to centralize data.

Most of the solutions allow centralized control [Zhao2011] [Bouajila2012] [Choi2012] [Lui2013] [Ituero2012], with [Bouajila2012] also permitting distributed control and information aggregation. When addressing large platforms, these are broken into subsystems (in some works referred to as islands), with each subsystem having its own interconnect and having its own monitoring infrastructure. For example, for large systems, the authors of [Bouajila2012] divided the SoC on islands and use a customized ring for each island. Inter-subsystem messages can be exchanged by means of interconnect on top of the aforementioned one [Bouajila2012] [Madduri2009] [Zhao2011]. This type of solution is also adopted in this work.

Sensor/monitor architecture is typically made of two parts: interface with the choice of interconnect, and processing part, where the data is being aggregated/actuated (sensing application dependent). Furthermore, on the processing side (which is connected to the system being monitored), the coupling between the sensors and the target system can be classified in two categories: tight (when actual instrumentation of the component design is required for introducing event counters and additional ports for reporting their activity [Lui2013]), and weak (when already built-in support exists – e.g. thermal sensors, performance counters, or some sort of non-intrusive way of accounting events is proposed [Madduri2009] [Zhao2011] [Ituero2012]).

Previous work [Zeng2002] [Snowdon2009] [Roy2011] on energy accounting in the OS layer relies on approximating per-thread energy consumption through modeling instead of direct measurements. Specifically, the OS leverages per-core performance monitoring counters and coarse-grain power consumption sensors on selected hardware components, such as the CPU socket or memory, to assign energy consumption to each executing thread. However, this approach may suffer from modeling inaccuracies. More importantly, it cannot accurately account for energy consumption on shared components, when accessed by more than one thread, such as the memory subsystem or buses. By contrast, our LEM approach enables accurate accounting at the granularity of individual hardware components, for both private and shared resources.

PTEM is discussed only by few papers [Lui2013] [Molka2010], most of them targeting x86 processor system architectures [Molka2010]. In [Lui2013] PTEM is achieved on the basis of resource utilization and occupancy tracking with dedicated hardware. Considering the state of the art, most of the recent efforts in energy accounting are directed toward component level energy modelling, profiling, or monitoring. However, energy consumed by data movement is not taken in consideration, due to its complexity and shared usage problem. In our work we address energy accounting of data movement in a heterogeneous multi-core system in conjunction with energy accounting of processing elements.

This work contributes to the state-of-the art by proposing a versatile low cost monitoring infrastructure with the software drivers that provide incentive for PTEM at OS level. Furthermore, it advances a novel scheduling concept – that of energy interrupt, that can bring the energy efficient processing in the foreground. Last, but not least, we have tested modern FPGA capabilities for measuring/monitoring energy consumption and compared our results against simulation-based switching activity measurements for ASIC technologies. In order to validate all these, we have implemented a multi-core system on a Zynq FPGA board.

7.5 SUPPORTING GRANTS, RESEARCH TEAM AND SCIENTIFIC OUTCOMES

Project title: GreenEr Mobile Systems by Cross LAyer Integrated energy Management

National competition: CHIST-ERA

Implementation period: 2012-2015

Project acronym: GEMSCLAIM

Project code: CHIST-ERA/1/01.10.2012

Budget: 250 000 EUR

Project summary: The GEMSCLAIM project aims at introducing novel approaches for reducing the “greed for energy” of modern battery powered systems, thereby improving the user experience and enabling new opportunities for mobile computing. Mobile terminals and consumer devices are among the fastest growing markets in computing. In the long term, further growth is endangered by the “power/ energy wall”. The purpose of GEMSCLAIM is to explore new techniques in energy optimization via an interdisciplinary vertical approach: a novel combined optimization across the major HW/SW system layers (compiler/OS/HW platform).

The ever-growing need for energy efficient computation requires adequate support for energy-aware thread scheduling that offers insight into a systems behavior for improved application energy/performance optimizations. Runtime accurate monitoring of energy consumed by every component of a multi-core embedded system is an important feature to be considered for future designs. Although, important steps have been made in this direction, the problem of distributing energy consumption among threads executed on different cores for shared components remains an ongoing struggle. We aim at designing a generic low-cost and energy efficient hardware

infrastructure which supports thread level energy consumption monitoring of hardware components in a multi-core system. The proposed infrastructure provides upper layers (operating system and application threads) with per thread and per component energy accounting API (Application Programming Interface), similar with performance profiling functions. Implementation results indicate that the proposed LEM (Load and Energy Monitor) adds around 10% resource overhead to the monitored system. Regarding the power estimates, the one derived by LEM achieve a correlation degree of more than 95% with the ones obtained from physical power measurements.

The research team involved in the project included experienced researchers, young researchers, external researchers and students:

PhD. Marius MARCU (local partner director)
PhD. Oana AMARICAI-BONCALO
PhD. Alexandru AMARICAI-BONCALO
PhD. Cosmin CERNAZANU-GLAVAN
PhD. Sebastian FUICU
PhD. Razvan BOGDAN
PhD. Gabriel GIRBAN
Eng. Lucian BARA
Eng. Madalin GHENEA
Eng. Marian IONASCU

Partners:

Innsbruck University (Lead partner)
Queen's University Belfast
RWTH Aachen University

The contributions of this work are as follows:

- hardware infrastructure for dynamic energy consumption monitoring in a heterogeneous multi-core system with per-thread energy accounting;
- energy interrupt specification and design;
- a use case on the software side (OS and drivers) for run-time per-thread energy accounting implementation on FPGA; and
- validation of proposed infrastructure on a high-end FPGA board with physical energy measurements.

This section is not published yet, but it has been submitted to ARC2016 conference and presented to one of the HiPEAC workshops in January 2015.

7.6 CONCLUSIONS AND FUTURE WORK

In this section, we have introduced a cost effective LEM infrastructure for component level power and energy monitoring by providing adequate hardware and software support for PTEA and energy

interrupt. The monitoring infrastructure implements two levels of energy accounting: processing energy and data movement energy. Per core energy accounting can be done using the LEM hardware infrastructure. The infrastructure can be further used in conjunction with OS drivers in order to, to implement thread-level energy accounting at OS level.

We have validated our infrastructure on a Zynq ZC702 evaluation board. We have developed systems consisting of 1 MB core, 2 MB cores, 4 MB cores and 8 MB cores. The results from the execution of WCET benchmarks has indicated a strong correlation between the LEM based energy estimates and the physical power board measurements of more than 95%. The implementation results indicated that the overall overhead of the proposed infrastructure is around 10%, for 14 sensors attached to 4-cores reference design. The proposed LEM has lower cost with respect to the Xilinx based performance counters, while having increased flexibility and accuracy.

8 SCIENTIFIC AND ACADEMIC DEVELOPMENT PLAN

The main research topics I plan to address on medium and long term will further develop the preoccupations and research interests I had since I finished the PhD studies, as follows:

- Power and Energy Measurement, Modeling, Optimization and Low-Power Design of Embedded Systems
- Dynamic Thermal, Power and Energy Management of Computing Systems
- Microprocessor Systems Design and Microprocessor Architectures
- Computer Networks and Wireless Communication
- Mobile Computing, Systems and Applications
- Low Level Software and Drivers Development

However the topics and problems addressed within all these domains become too broad, requiring very high and narrow specialization in order to perform well at the European level. On the other hand, deep specialization will limit the opportunities and the evolution of the research team. Therefore, in order to increase our research impact and expertise, I plan on short term, to focus and consolidate my core expertize that is:

- Energy measurements, energy profiling an energy efficiency analysis of any kind of computing or electronic device with high accuracy and at high sampling rates.

The core research topic is of extreme actuality, being applicable to any kind of device, starting with ordinary desktop and laptop computers and ending with smart devices (watches or mobiles) and high end cloud and server infrastructures. In the near future, higher data rates and traffic, advanced media codecs, and graphics applications will ask for even more energy than the battery can deliver. At the same time, the power density limit might lead to a significant share of "Dark Silicon" at 22nm CMOS and below. Obviously, disruptive energy optimizations are required that go well beyond traditional technologies like DVFS (dynamic voltage and frequency scaling) and power-down of temporarily unused components. Therefore, highly accurate and efficient energy accounting and measurement is required by any modern computing system.

On the other hand, the problem I faced with in my core domain of expertise is that it is mainly oriented on analyzing existing hardware and software solutions and identifying design or implementation problems based on energy and power signatures or profiles, lacking any optimization solution or design enhancement. Therefore, the second research topic I will focus on short term will be directed toward power optimization and increasing the energy of the hardware and software solutions. However, this is not an easy task because it is highly linked to the architectures and technologies used, such that no general optimization solution is possible. Hence, based on previous experience, and further refining the goals of the second research direction, it can be defined as:

- Context-aware cross-layer aggressive optimization of heterogeneous multicore battery powered devices (embedded and mobile)

Lastly, but not last, considering that the computing domain is very dynamic, so that new opportunities arising at a fast pace can easily increase the visibility and impact of the research team, I will continuously try to identify new innovative applications for our research outcomes and find disrupting research topics in the field of mobile systems and applications. Smart technologies are currently one of the most promising domain both for research and innovation. We can target with mobile applications directly a huge number of users on the global market with less effort through the existing application markets (e.g. Google Play, Apple Store, Windows Store or Amazon Store). This market will let us test new kind of applications in order to bring new innovative products on the market. Therefore, as the third applicative research direction I will try to find new collaborations and resources in the field of:

- User-centric smart technologies and applications as a glue between mobile devices and Internet of Things

With the last research direction, that is the most applicative and market oriented one, I will try to address the whole life cycle of a new product (no matter it is a research outcome or consumer one), starting from the idea and ending with a technology transfer to a company or product development within a startup or spinoff company. I plan to address the whole applicative research life cycle together with our students and companies in Timisoara, following several phases:

- Find and recruit good students from early years of study (2nd or 3rd grade) by involving them in projects and targeting students' competitions, such as ImagineCup competition. Our team has vast experience in addressing such competitions starting from 2006 with good results in 2009 (selected in top 15 teams worldwide), 2010 (top 6 worldwide), 2012 (won the national phase and selected in top 20 teams worldwide), 2014 (won the national phase). We are looking for good student not only with technical skills but also to their presentation skills, marketing and managerial skills. We are investing also in training them for these competitions.
- Invite students to attend Mobile Systems and Applications classes – this is an important step because here they will learn not only technical knowledge on mobile applications development, but also we address the market perspective (how to advertise your apps on mobile markets, economical perspective (how to monetize your apps), intellectual property (how to protect your work), evaluate an idea of a product. The student are encouraged and actively supported by our team to develop their own product and launch it on the market during the semester.
- Keep the selected students to work on our research projects during the diploma thesis implementation in the last year of study. Each year in our team we coordinate between 10-20 students, investing a lot of effort to achieve good implementations.

- Identify good diploma projects implementations to be further developed during the two years of master studies (the research oriented projects) or identify a company interested in the outcome (the product oriented projects).
- Identify research oriented students based on previous collaboration during master studies in order to continue their research studies at PhD level and prepare for them positions in our financed grants. Some of the students in this category start to work from the master level.
- Identify financing sources (companies, structural funds) to help students having promising product prototypes, to develop the prototype into a product and start a company.
- Look for collaboration with the local industry, especially the ones having Romanian investors, to transfer the research outcome toward the interested companies in order to develop new products or services.
- Look for local or external funds to continue our research projects. Despite the high competition and lack of research fund we managed in our team to win more than 7 projects in last 5 years.

We already experienced the previous steps with some encouraging results:

- Each year, the selected projects developed at Mobile Systems and Applications are presented in a workshop where we invite people from industry to evaluate the prototypes. Several applications are also published on the mobile apps markets (2-5 new apps each year). One of the application published in 2013 achieved more than 50000 downloads at the beginning of 2015.
- We started a company in 2014 to develop a product using the results achieved in collaboration with a student during diploma thesis (2009) and master dissertation (2011). The company has been granted with ~200000 EUR from structural founds POS-CCE program.

But we are also facing with several challenges. The most important one is the human resource market in Timisoara that is overwhelmed by the increasing demands of a large number of software and automotive companies in the region. Therefore, we have to compete on our students with big companies. We address this problem with our proposed research life cycle, which we hope that will show the results on long term. The second challenge is to speed-up the process. We are encouraged by the experience we had with the startup company, waiting to see the results in 2016, after a collaboration of more than 7 years. This is our first experience in following the phases, but it was quite a long process, that had to be improved in order to achieve success on the dynamic market today.

9 REFERENCES

9.1 RELEVANT PUBLISHED PAPERS

[1] Cosmin Cernazanu, **Marius Marcu**, Anomaly Detection Using Power Signature of Consumer Electrical Devices, Advances in Electrical and Computer Engineering Journal, 15 (1), pp. 89-94, 2015, ISSN: 1582-7445, IF2015: 0.529

[2] Cosmin Cernazanu, **Marius Marcu**, Alexandru Amaricai, Stefan Fedeac, Madalin Ghenea, Zheng Wang, Anupam Chattopadhyay, Jan Weinstock, Rainer Leupers, Direct FPGA-based Power Profiling for a RISC Processor, IEEE International Instrumentation and Measurement Technology Conference (I2MTC 2015), Mai 2015, Pisa, Italia

[3] Stefan Fedeac, **Marius Marcu**, Cosmin Cernazanu, Alexandru Amaricai, Energy Profiling of FPGA Designs, 2014 IEEE Symposium on Robotic and Sensors Environments (ROSE2014), Oct. 2014, Timisoara, Romania

[4] **Marius Marcu**, Power–thermal profiling of software applications, Microelectronics Journal, 42 (4), pp.601-608, Apr. 2011, ISSN 0026-2692, IF2015: 0.836

[5] **Marius Marcu**, Dacian Tudor, Energy Consumption Model for Mobile Wireless Communication, Proceedings of the 9th ACM international symposium on Mobility management and wireless access (MobiWac 2011), pp.91-94, Oct. 2011, Miami, Florida, ISBN: 978-1-4503-0901-1

[6] **Marius Marcu**, Dacian Tudor, Power Consumption Measurements of Virtual Machines, Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2011), pp.445-449, May 2011, Timisoara, Romania, Print ISBN: 978-1-4244-9108-7

[7] **Marcu Marius**, Tudor Dacian, Moldovan Horatiu, Fuicu Sebastian, Popa Mircea, Energy Characterization of Mobile Devices and Applications Using Power-Thermal Benchmarks, Microelectronics Journal, 40 (7), pp.1141-1153, Jul. 2009, ISSN 0026-2692, IF2015: 0.836

[8] Tudor Dacian, **Marcu Marius**, Designing a Power Efficiency Framework for Battery Powered Systems, Proceedings of Israeli Experimental Systems Conference (SYSTOR 2009), pp.1-8, May 2009, Haifa, Israel, ISBN 978-1-60558-623-6

[9] **Marius Marcu**, Dacian Tudor, Sebastian Fuicu, Horatiu Moldovan, and Voicu Groza, A View on Mobile Terminal Power Efficiency of Wireless Communication, Proceedings of the 25th IEEE International Instrumentation and Measurement Technology Conference (I2MTC 2008), pp.382-387, May. 2008, Victoria, Vancouver Island, Canada, ISBN 978-1-4244-1540-3

[10] **Marius Marcu**, Mircea Vladutiu, Horatiu Moldovan, Mircea Popa, Thermal Benchmark and Power Benchmark Software, Proceedings of the 12th IEEE International Workshops on THERMal Investigations of ICs and Systems (THERMINIC 2006), pp.203-208, Sep. 2006, Nice, France, ISBN 2-916187-0409

9.2 RELATED WORK PAPERS

[Iyenggar2007] A. Iyenggar, A. Tripathi, A. Basarur and I. Roy, "Unified Power Management Framework for Portable Media Devices", IEEE International Conference on Portable Information Devices, PORTABLE07, May 2007, Orlando, USA.

[Chen2008] B. Chen, W. Pak Tu Ma, Y. Tan, A. Fedorova, and G. Mori, "GreenRT: A Framework for the Design of Power-Aware Soft Real-Time Applications", Workshop on the Interaction between Operating Systems and Computer Architecture, WIOSCA2008, 2008, Beijing, China.

[Fei2008] Y. Fei, L. Zhong, and N. K. Jha, "An Energy-Aware Framework for Dynamic Software Management in Mobile Computing Systems", ACM Transactions on Embedded Computing Systems Journal, Vol. 7, Iss. 3, pp. 27-58, April 2008.

[Cheung2009] T. L. Cheung, K. Okamoto, F. Maker, X. Liu, and V. Akella, "Markov Decision Process (MDP) Framework for Optimizing Software on Mobile Phones", Proceedings of the seventh ACM international conference on Embedded software, EMSOFT '09, 2009, New York, USA.

[Bellasi2010] P. Bellasi, S. Bosisio, M. Carnevali, W. Fornaciari, and D. Siorpaes, "Constrained Power Management: Application to a Multimedia Mobile Platform", Proceedings of the Conference on Design, Automation and Test in Europe, DATE'10, Leuven, Belgium.

[Vallina2011] N. Vallina-Rodriguez and J. Crowcroft, "ErdOS: Achieving Energy Savings in Mobile OS", Proceedings of the 6th ACM International Workshop on Mobility in the Evolving Internet Architecture, MobiArch'11, June 28, 2011, Bethesda, Maryland, USA.

[Mittal2012] R. Mittal, A. Kansal, and R. Chandra, "Empowering Developers to Estimate App Energy Consumption", Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Mobicom2012, 2012, New York, USA.

[Krintz2004] C. Krintz, Y. Wen, and R. Wolski, "Application-level Prediction of Battery Dissipation", ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), pp224-229, August 9-11, 2004, Newport Beach, CA.

[Agarwal2007] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin and R. Gupta, "Wireless Wakeups Revisited: Energy Management for VoIP over Wi-Fi Smartphones", MobiSys'07, June 11-14, 2007, San Juan, USA.

[Feeney2001] L. M. Feeney and M. Nilsson. "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment". In IEEE INFOCOM 2001, 2001.

[Krashinsky2005] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown", Wireless Networks 11, 135–148, 2005.

[Carpenter2007] R.E. Carpenter, "Comparing Multi-Core Processors for Server Virtualization", White Paper, Intel Information Technology. http://www.multicoreinfo.com/research/papers/whitepapers/multicore_virtualization.pdf, 2007.

[Dhiman2009] G. Dhiman, G. Marchetti, and T. Rosing, "vGreen: A System for Energy Efficient Computing in Virtualized Environments", International Symposium on Low Power Electronics and Design, ISLPED'09, USA, 2009.

[Clark2005] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines", Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, NSDI'05, 2005.

[Tian2007] K. Tian, K. Yu, J. Nakajima, and W. Wang, "How virtualization makes power management different", Proceedings of the Linux Symposium, OLS'07, Canada, 2007.

[Behmann2009] F. Behmann, "Virtualization for Embedded Power Architecture CPS", Electronics Products, 2009.

[Chetan2010] S. Chetan, G. Kumar, K. Dinesh, K. Mathew and M.A. Abhimanyu, "Cloud Computing for Mobile World", Research work, http://chetan.ueuo.com/projects/CCMW.pdf (2010)

[Nathuji2007] R. Nathuji and K. Schwan, "VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems", ACM SIGOPS Operating Systems Review, Vol.41 (6), SOSP'07, 2007.

[Verma2008] A. Verma, P. Ahuja, and P. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems", Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, Leuven, Belgium, Moddleware'08, 2008.

[Ye2010] L. Ye, G. Lu, S. Kumar, C. Gniady, and J.H. Hartman, "Energy-Efficient Storage in Virtual Machine Environments", Proceedings of the 6th ACM SIGPLAN International Conference on Virtual Execution Environments, Pittsburgh, USA, VEE'10, 2010.

[Hartig2008] H. Härtig, M. Roitzsch, A. Lackorzynski, B. Döbel, and A. Böttcher, "L4-Virtualization and Beyond", Korean Information Science Society Review, December 2008, (2008)

[Fehrenbacher2009] K. Fehrenbacher, "10 Monitoring Tools Bringing Smart Energy Home", Business Week, Apr. 2009, http://www.businessweek.com/technology/content/apr2009/tc20090414_446611.htm.

[LeBlanc2007] J. LeBlanc, "Device-Level Power Consumption Monitoring", Proceedings of the 9th International Conference on Ubiquitous Computing, Sep. 2007, Austria, http://www.awarepower.com/ ubicomp07.pdf.

[Berges2009] M. Berges, E. Goldman, H. S. Matthews, L. Soibelman, "Learning Systems for Electric Consumption of Buildings", Proceedings of the ASCE International Workshop on Computing in Civil Engineering, Austin, Texas, 2009.

[Adamo2007] F. Adamo, F. Attivissiomo, G. Cavone, and A.M. Lanzolla, "A Virtual Instrument for the Electric Power Monitoring in the Distributing Network", 15th Symposium on Novelties in Electrical Measurements and Instrumentation, Romania, 2007.

[Crosbie2008] T. Crosbie, "Household energy consumption and consumer electronics: The case of television", Energy Policy vol. 36, pp. 2191-2199, Apr. 2008.

[McAllister2007] J. A. McAllister and A. E. Farrell, "Electricity consumption by battery-powered consumer electronics: A household-level survey", Energy vol. 32 pp. 1177–1184, 2007.

[Elias2009] E.W.A. Elias, E.A. Dekoninck, and S.J. Culley, "Quantifying the Energy Impacts of Use: A Product Energy Profile Approach", in Proc. 16th CIRP International Conference on Life Cycle Engineering, 4-6 May 2009, Cairo, Egypt.

[Froehlich2011] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. S. Reynolds, and S. N. Patel, "Disaggregated End-Use Energy Sensing for the Smart Grid", IEEE Pervasive Computing, Special Issue on Smart Energy Systems, Jan–Mar 2011.

[Ou2012] Q. Ou, Y. Zhen, X. Li, Y. Zhang, L. Zeng; , "Application of Internet of Things in Smart Grid Power Transmission", Mobile, Ubiquitous, and Intelligent Computing (MUSIC), 2012 Third FTRA International Conference on , pp.96-100, 26-28 June 2012

[Huang2011] S. J. Huang, C. T. Hsieh, L. K. Kuo, C. W. Lin, C. W. Chang, S. A. Fang, "Classification of home appliance electricity consumption using power signature and harmonic features", Power Electronics and Drive Systems (PEDS), 2011 IEEE Ninth International Conference on , pp.596-599, 5-8 Dec. 2011

[Lee2012] W. K. Lee, G. S. K. Fung, H. Y. Lam, F. H. Y. Chan, and M. Lucente, "Exploration on Load Signatures", International Conference on Electrical Engineering, ICEE 2004, Sapporo, Japan, Jul. 2012.

[Drif2014] M. Drif, A. J. M. Cardoso, "Stator Fault Diagnostics in Squirrel Cage Three-Phase Induction Motor Drives Using the Instantaneous Active and Reactive Power Signature Analyses", Industrial Informatics, IEEE Transactions on , vol.10, no.2, pp.1348-1360, May 2014

[Hassan2014] T. Hassan, F. Javed, N. Arshad, "An Empirical Investigation of V-I Trajectory Based Load Signatures for Non-Intrusive Load Monitoring", Smart Grid, IEEE Transactions on, vol.5, no.2, pp. 870-878, March 2014

[Abu-Siada2013] A. Abu-Siada, N. Hashemnia, S. Islam, M. Masoum, "Understanding power transformer frequency response analysis signatures", Electrical Insulation Magazine, IEEE , vol.29, no.3, pp. 48-56, May 2013

[Jiang2009] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler, "Design and Implementation of a High-Fidelity AC Metering Network", The 8th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN'09, 2009, San Francisco, California, USA.

[Patel2007] S. Patel, T. Robertson, J. Kientz, M. Reynolds, and G. Abowd. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In UbiComp 2007: Ubiquitous Computing, LNCS 4717, Springer, pages 271–288, 2007.

[Farinaccio1999] R. Z. L. Farinaccio. Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses. In Energy and Buildings, 1999.

[Marceau2000] M. L. Marceau and R. Zmeureanu. Nonintrusive load disaggregation computer program to estimate the energy consumption of major end uses in residential buildings. In Energy Conversion and Management, volume 41, pages 1389 – 1403, 2000.

[Armenia2010] A. Armenia and J. H. Chow. A flexible phasor data concentrator design leveraging existing software technologies. IEEE Transactions on Smart Grid, 1(1):73–81, 2010.

[Zhang2010] Y. Zhang, P. Markham, T. Xia, L. Chen, Y. Ye, Z. Wu, Z. Yuan, L. Wang, J. Bank, J. Burgett, R. W. Conners, and Y. Liu. Wide-area frequency monitoring network (FNET) architecture and applications. IEEE Transcations on Smart Grid, 1(2):159–167, 2010.

[Lim2006] H. Lim, L. Kung, J.C. Hou, H. Luo, Zero-Configuration, Robust Indoor Localization: Theory and Experimentation, INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, pp. 1-12, 2006.

[Grosmann2006] U. Grosmann, C. Rohrig, S. Hakobyan, T. Domin, and M. Dalhaus, "WLAN indoor positioning based on Euclidian distance and interpolation", Proceedings of the 8th Wireless Technologies Kongress, pp. 296-305, Dortmund, Germany, 2006.

[Broxton2005] M. Broxton, J. Lifton, J. Paradiso, "Localizing a Sensor Network via Collaborative Processing of Global Stimuli", Proceedings of the Second European Workshop on Wireless Sensor Networks, pp. 321-332, 2005.

[Ali2005] A.A. Ali and A.S. Omar, "Time of Arrival Estimation for WLAN Indoor Positioning Systems using Matrix Pencil Super Resolution Algorithm", Proceedings of the 2nd Workshop on Positioning, Navigation and Communication, WPNC'05, pp. 11-20, 2005.

[Zhang2006] C. Zhang , M.J. Kuhn, M. Brandon, A.E. Fathy, M. Mahfouz, "High Resolution Uwb Indoor Localization System Operating At 8 and 24 GHz Utilizing Time Difference Of Arrival Approach", National Radio Science Meeting, 2006.

[Li2004] X. Li, K. Pahlavan, "Super-Resolution TOA Estimation With Diversity for Indoor Geolocation," IEEE Transactions on Wireless Communications vol.3 No.1., pp. 224-234, 2004.

[Chen2002] J.C. Chen, K. Yao, R.E. Hudson, "Source localization and beamforming", IEEE Signal Processing Magazine, Vol. 19, No. 2, 2002, pp. 30-39.

[Priyantha2000] N.B. Priyantha, A. Chakraborty, H. Balakrishnan, "The Cricket Location-Support System", Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (ACM MOBICOM), 2000, pp.32-43.

[Retscher2006] G. Retscher, E. Moser, D. Vredeveld and D. Heberling, "Performance and Accuracy Test of the WLAN Indoor Positioning System ipos", Proc. of the 3rd Workshop on Positioning, navigation and Communication (WPNC'06), pp. 7-14, 2006.

[Li2005] B. Li, Y. Wang, H.K. Lee, A.G. Dempster, C. Rizos, "A new method for yielding a database of location fingerprints in WLAN", IEE Proceedings Communications, Vol. 152, No. 5, 2005, pp. 580-586.

[Salter2006] B. Li, J. Salter, A.G. Dempster. C. Rizos, "Indoor positioning techniques based on Wireless LAN", First IEEE Int. Conf. on Wireless Broadband & Ultra-Wideband Communications, 2006.

[Grossmann2007] U. Grossmann, M. Schauch, S. Hakobyan, "The accuracy of algorithms for WLAN indoor positioning and the standardization of signal reception for different mobile devices", International Journal of Computing, vol. 6, no. 1, pp. 103-109, 2007.

[Yiming2006] J. Yiming, S. Biaz, S. Pandey, P. Agrawal, "ARIADNE: A Dynamic Indoor Signal Map Construction and Localization System", Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, 2006, pp. 151-164.

[Bahl2000] P. Bahl, V.N. Padmanabhan, „RADAR: An In-Building RF-Based User Location and Tracking System", INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol.2, 2000, pp.775-784.

[Karakos2005] A. Karakos, I. Kouris, "Wireless positioning in a university library", WSEAS Transactions on Circuits and Systems. Vol. 4, no. 7, pp. 455-462. July 2005.

[Tong2008] J.G. Tong, M.A.S. Khalid, "Profiling Tools for FPGA-Based Embedded Systems: Survey and Quantitative Comparison", published in Journal of Computers, Vol.3, No. 6(2008), pp. 1-14, June 2008.

[Aldham2011] M. Aldham, J. Anderson, S. Brown, A. Canis, "LowCost Hardware Profiling of Run-Time and Energy in FPGA Embedded Processors", published in 2011 IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp.61-68, September 2011

[Genser2009] A. Genser, C. Bachmann, J. Haid, C. Steger, R. Weiss, "An Emulation-Based Real-Time Power Profiling Unit for Embedded Software", published in International Symposium on Systems, Architectures, Modeling, and Simulation SAMOS '09, pp. 67-73, July 2009.

[Powell2013] A. Powell, C. Savvas-Bouganis, Peter Y.K. Cheung, "High-level power and performance estimation of FPGA-based soft processors and its application to design space exploration", published in Journal of Systems Architecture: the EUROMICRO Journal, Vol. 59, Issue 10, November 2013, pp. 1144-1156.

[Laopoulos2003] Th. Laopoulos, P. Neofotistos, C. Kosmatopoulos, S. Nikolaidis, "Measurement of Current Variations for the Estimation of Software-related Power Consumption", published in IEEE Transactions on Instrumentation and Measurement, Vol.52, Issue 4, Aug. 2003.

[Kavvadias2004] N. Kavvadias, P. Neofotistos, S. Nikolaidis, K. Kosmatopoulos and Th. Laopoulos "Measurements Analysis of the Software-Related Power Consumption in Microprocessors", published in IEEE Transactions on Instrumentation and Measurement, Vol. 53, Issue 4, Aug. 2004.

[Nakutis2009] Z. Nakutis, "Embedded Systems Power Consumption Measurement Methods Overview", published in MATAVIMAI Academic Journal; Measurements, Vol. 44, Issue 2, pp. 29-35, Kaunas, 2009.

[Beldachi2014] A. F. Beldachi, J. L. Nunez-Yanez, "Accurate Power control and monitoring in ZYNQ boards", 24th International Conference on Field Programmable Logic and Applications (FPL), Sep. 2014

[Schurmans2013] S. Schürmans, Diandian Zhang, Dominik Auras, Rainer Leupers, Gerd Ascheid, Creation of ESL Power Models for Communication Architectures using Automatic Calibration, 50th ACM / EDAC / IEEE Design Automation Conference (DAC), Jun. 2013

[Ou2004] J. Ou, and V.K. Prasanna, "Rapid Energy Estimation of Computations on FPGA based Soft Processors". In Proceedings of the IEEE International SOC Conference, (Santa Clara, California, September 12-15, 2004). DOI=http://10.1109/SOCC.2004.1362437

[Afifi2014] S.M. Afifi, F. Verdier, C. Belleudy, "Power Estimation Method Based on Real Measurements for Processor-Based Designs on FPGA", Computational Science and Computational Intelligence (CSCI), 2014 International Conference on , vol.2, no., pp.260,263, 10-13 March 2014

[Pallares2012] V. Pallares-Lopez, A.M. Munoz, A. Gil-de-Castro, I. Santiago-Chiquero, "FPGA-based embedded system architecture for power quality measurements", 2012 IEEE 15th International Conference on Harmonics and Quality of Power (ICHQP), vol., no., pp.507,511, 17-20 June 2012

[Atitallah2013] R. Ben Atitallah, R.; E. Senn, D. Chillet, M. Lanoe, D. Blouin, "An Efficient Framework for Power-Aware Design of Heterogeneous MPSoC", IEEE Transactions on Industrial Informatics, vol.9, no.1, pp.487,501, Feb. 2013

[OpenCores2010] OpenCores, WISHBONE System-on-Chip (SoC) Interconnection, 2010.

[WCET] WCET benchmarks: http://www.mrtc.mdh.se/projects/wcet/benchmarks.html

[ARM2003] ARM, AMBA Open Specifications, http://www.arm.com/products/system-ip/amba/amba-open-specifications.php, 2003.

[Xilinx2014] Xilinx Inc., MicroBlaze Processor Reference Guide, UG984 (v2014.1) April 2, 2014.

[Weaver2012] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, "Measuring Energy and Power with PAPI", Proceedings of 41st International Conference on Parallel Processing Workshops, ICPPW'12, September 2012, Pittsburgh, PA, USA.

[Bouajila2012] A. Bouajila, A. Lakhtel, J. Zeppenfeld, W. Stechele, and A. Herkersdorf, "A low-overhead Monitoring Ring Interconnect for MPSoC Parameter Optimization", Proceedings of IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems DDECS2012, Tallinn, Estonia, April 2012.

[Madduri2009] S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, "A Monitor Interconnect and Support Subsystem for Multicore Processors", Proceedings of Design, Automation & Test Europe, DATE2009, Nice, France, April 2009.

[Zhao2011] J. Zhao, S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, "A Dedicated Monitoring Infrastructure for Multicore Processors", IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 19, No. 6, June 2011.

[Choi2012] S. Choi, H. Hwang, B. Song, and H. Cha, "Hardware-assisted energy monitoring architecture for micro sensor nodes", Journal on System Architecture, Elsevier, No 58, pp 73-85, 2012.

[Lui2013] Q. Lui, M. Moreto, V. Jimenez, J. Abella, F.J. Cazorla, and M. Valero, "Hardware Support for Accurate Per-Task Energy Metering in Multicore Systems", ACM Transactions on Architecture and Code Optimization, Vol. 10, No. 4, December 2013.

[Ituero2012] P. Ituero, M. López-Vallejo, M.A.S. Marcos, and C.G. Osuna, "Light-Weight On-Chip Monitoring Network for Dynamic Adaptation and Calibration", IEEE Sensors Journal, Vol. 12, No. 6, June 2012.

[Molka2010] D. Molka, D. Hackenberg, R. Schone, and M.S. Millier, "Characterizing the Energy Consumption of Data Transfers and Arithmetic Operations on x86-64 Processors", In Proceedings of International Green Computing Conference, GreenComp2010, Chicago, USA, August 2010.

[Zeng2002] H. Zeng, C.S. Ellis, A.R. Lebeck, and A. Vahdat. "ECOSystem: managing energy as a first class operating system resource", In Proceedings of the 10th international conference on Architectural support for programming languages and operating systems (ASPLOS X). ACM, New York, NY, USA, pp.123-132, 2002.

[Snowdon2009] D.C. Snowdon, E. Le Sueur, S.M. Petters, and G. Heiser, "Koala: a platform for OS-level power management", In Proceedings of the 4th ACM European conference on Computer systems (EuroSys'09), New York, NY, USA, pp. 289-302, 2009.

[Roy2011] A. Roy, S.M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich, “Energy management in mobile devices with the cinder operating system”, In Proceedings of the sixth conference on Computer systems (EuroSys '11). ACM, New York, NY, USA, pp. 139-152, 2011.