Microprocessors and Microsystems 000 (2016) 1–9

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

ENTRA: Whole-systems energy transparency

Kerstin Eder[a], John P. Gallagher[b,c,*], Pedro López-García[c,e], Henk Muller[d], Zorana Banković[c], Kyriakos Georgiou[a], Rémy Haemmerlé[c], Manuel V. Hermenegildo[c,f], Bishoksan Kafle[b], Steve Kerrison[a], Maja Kirkeby[b], Maximiliano Klemen[c], Xueliang Li[b], Umer Liqat[c], Jeremy Morse[a], Morten Rhiger[b], Mads Rosendahl[b]

[a] University of Bristol, United Kingdom
[b] Roskilde University, Denmark
[c] IMDEA Software Institute, Spain
[d] XMOS Ltd., Bristol, United Kingdom
[e] Spanish Council for Scientific Research, Spain
[f] Technical University of Madrid, Spain

ARTICLE INFO

Article history:
Received 13 January 2016
Revised 27 May 2016
Accepted 8 July 2016
Available online xxx

Keywords:
Energy transparency
Energy-aware software development
Energy modelling
Static analysis
Resource analysis

ABSTRACT

Promoting energy efficiency to a first class system design goal is an important research challenge. Although more energy-efficient hardware can be designed, it is software that controls the hardware; for a given system the potential for energy savings is likely to be much greater at the higher levels of abstraction in the system stack. Thus the greatest savings are expected from energy-aware software development, which is the vision of the EU ENTRA project. This article presents the concept of *energy transparency* as a foundation for energy-aware software development. We show how energy modelling of hardware is combined with static analysis to allow the programmer to understand the energy consumption of a program without executing it, thus enabling exploration of the design space taking energy into consideration. The paper concludes by summarising the current and future challenges identified in the ENTRA project.

1. Introduction

Energy efficiency is a major concern in systems engineering. The EU's Future and Emerging Technologies MINECC programme aims to "lay the foundations for radically new technologies for computation that strive for the theoretical limits in energy consumption." The research objectives range from physics to software; they include, among others, new elementary devices, as well as "software models and programming methods supporting the strive for the energetic limit." The ENTRA project, http://entraproject.eu, addresses the latter objective; we focus on energy transparency, which we regard as a key prerequisite for new energy-aware system development methods and tools.

The ENTRA project ran from October 2012 to December 2015 (39 months) and was funded by the European Commission under the 7th Framework Programme. The consortium contained three research institutions and one industrial partner specialising in the design of advanced multicore microcontrollers (XMOS xCORE). The overview of the project structure is shown in Fig. 1. The foundations for the central concept of energy transparency were developed in two work packages (WP2 and WP3) on energy modelling and energy analysis respectively. Energy transparency enables energy optimisations, studied in WP4. WP1 concerned the development of tools and techniques applicable in energy-aware software development. Finally there were work packages dealing with benchmarking, evaluation, dissemination and project management. This paper summarises mainly the outcomes of work packages WP1, WP2, WP3, WP4 and WP6. The public deliverables of the project are all available on the project website `http://entraproject.eu`.

After this introduction, we discuss the two main areas of research supporting energy transparency. Section 2 presents approaches for building models of software energy consumption at different levels of abstraction. Section 3 contains an overview of static resource analysis techniques, showing how an energy model can be used in analysis of a program's energy consumption. Section 4 summarises the role of energy transparency in energy-aware software development, discusses the achievements in the project so far, and outlines current challenges and directions for future research.

* Corresponding author.
E-mail address: jpg@ruc.dk (J.P. Gallagher).

http://dx.doi.org/10.1016/j.micpro.2016.07.003
0141-9331/

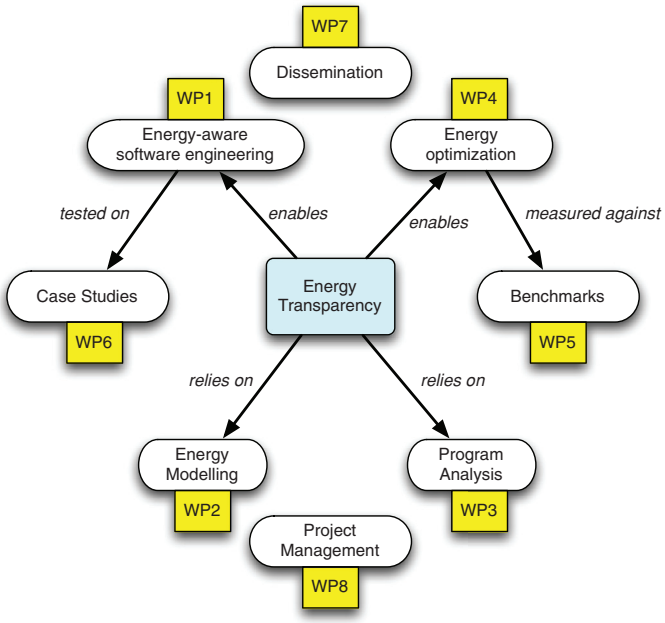

Fig. 1. Overview of the ENTRA project work plan.

1.1. Energy-aware computing

Energy-aware computing is a research challenge that requires investigating the entire system stack from application software and algorithms, via programming languages, compilers, instruction sets and micro architectures, to the design and manufacture of the hardware. This is because energy is consumed by the hardware performing computations, but the control over the computation ultimately lies within the applications running on the hardware. While hardware can be designed to save a modest amount of energy, the potential for savings is far greater at the higher levels of abstraction in the system stack. An estimate from Intel [1] is that energy-efficient software can realize savings of a factor of three to five beyond what can be achieved through energy efficient hardware. Roy and Johnson [2] list five objectives that "help make software design decisions consistent with the objectives of power minimization": match the algorithm to the hardware; minimize memory size and expensive memory accesses; optimise the performance, making maximum use of available parallelism; take advantage of hardware support for power management; and select instructions, sequence them, and order operations in a way that minimizes switching in the CPU and datapath. To achieve these objectives requires the programmer and/or the tools to understand the relationship between code and energy usage. Energy Transparency aims to enable exactly this.

1.2. Energy transparency

The concept of energy transparency is at odds with the trend in modern software engineering - the desire to abstract away machine-level details using high-level languages, abstract data types and classes, libraries and layers of interpretation or compilation, in the interests of portability, programmer productivity, understandability and software reuse. By contrast, energy transparency requires making visible how software impacts on energy consumption when executed on hardware. Availability of this information enables system designers to find the optimal trade-off between performance, accuracy, and energy usage of a computation. To achieve energy transparency, models of how energy is consumed during a computation are required. As will be discussed in Section 2, such models can be established at different levels of abstraction, ranging from models that characterize individual functional hardware blocks [3], via Instruction Set Architecture (ISA) characterization models [4–6], to models based on intermediate representations used by the compiler [7,8]. The final energy models provide information that feeds into static resource usage analysis algorithms [9–16], where they represent the energy usage of elementary parts of the computation. This is discussed in Section 3.

2. Energy modelling

Energy models can rely on information at several possible abstraction levels, from gate-level hardware description, through functional block and Instruction Set Architecture (ISA), up to performance counter or transaction based abstractions. Energy models at higher levels tend to be faster to use, but have lower accuracy than models at lower levels of abstraction. In ENTRA, the aim was to provide accurate modelling that can be exploited through analysis that is applied in order to estimate the energy consumed by software.

2.1. Defining and constructing an energy model

The ISA is a practical level of abstraction for energy modelling, because it expresses underlying hardware operations and their relationship with the intent of the software. Constructing a model at this level gives us the following benefits: energy costs can be attributed directly to individual machine instructions as output by the back end of the compiler; instruction properties and energy consumption are strongly correlated, e.g. energy consumption typically increases with increasing numbers of operands; and machine instructions can be traced back to the original source code statements written by the software developer, as well as to various intermediate representations.

However, energy modelling at the ISA level requires additional effort in order to produce useful models: instruction costs must be captured through a profiling suite and measurement of device power; in addition, indirect or statistical approaches are required to characterise instructions that cannot be profiled through direct measurements. Furthermore, for multi-threaded architectures other properties such as the cost of running multiple threads and the cost of idle periods must be determined.

Our target architecture for energy modelling and analysis is the XMOS xCORE embedded microcontroller [17]. Beyond offering timing-deterministic instruction execution, the xCORE is hardware multi-threaded and comes in a variety of multi-core configurations. The xCORE architecture is simple by design and, thus, ideal to investigate the advanced energy modelling and static analysis techniques required to achieve energy transparency. The techniques we developed are readily transferable to other deeply embedded, cache-less, IoT-type processors such as those in the ARM Cortex M series or the Atmel AVR. The fact that the xCORE offers multi-threading made it a particularly interesting target for the ENTRA project.

We have shown that in the xCORE the number of active threads has an impact upon energy consumption [6]. As such, the model must take this into account. Traditional ISA-level models, such as that of [18], can attribute energy costs simply to instructions, the transitions between instructions, and any additional effects that impact on energy consumption, such as cache hits and misses. Although we build on this principle, parallelism has to be considered, yielding a more complex model equation for E_p, the energy consumed by a program p:

$$E_p = P_b N_{idl} T_{clk} + \sum_{t=1}^{N_t} \sum_{i \in ISA} ((M_t P_i O + P_b) N_{i,t} T_{clk}) \quad (1)$$