Optimisation of Performance Metrics of Embedded Hard Real-Time Systems using Software/Hardware Parallelism

Antonio Paolillo

Summary

1 Introduction

Nowadays, embedded systems are part of our daily lives. Some of these systems are called *safety-critical* and have strong requirements in terms of safety and reliability. Additionally, these systems must have a long autonomy, good performance and minimal costs. Finally, these systems must exhibit predictable behaviour and provide their results within firm deadlines.

When these different constraints are combined in the requirement specifications of a modern product, classic design techniques making use of single core platforms are not sufficient. Academic research in the field of real-time embedded systems has produced numerous techniques to exploit the capabilities of modern hardware platforms. These techniques are often based on using parallelism inherently present in modern hardware to improve the system performance while reducing the platform power dissipation. However, very few systems existing on the market are using these state-of-the-art techniques. Moreover, few of these techniques have been validated in the context of practical experiments.

In the thesis, we realise the study of operating system level techniques allowing to exploit hardware parallelism through the implementation of parallel software in order to boost the performance of target applications and to reduce the overall system energy consumption while satisfying strict application timing requirements. We detail the theoretical foundations of the ideas applied in the dissertation and validate these ideas through experimental work. To this aim, we use a new Real-Time Operating System kernel written in the context of the creation of a spin-off of the Université libre de Bruxelles.

Our experiments are based on the execution of applications on the operating system which run on a real-world platform for embedded systems. Our results show that, compared to traditional design techniques, using parallel and power-aware scheduling techniques in order to exploit hardware and software parallelism allows to execute embedded applications with substantial savings in terms of energy consumption. We present future and ongoing research work that exploit the capabilities of recent embedded platforms. These platforms combine multi-core processors and reconfigurable hardware logic, allowing further improvements in performance and energy consumption.

2 State of the art and contributions

In the thesis dissertation, we tackle the problem of energy-efficient real-time embedded system design. Embedded system design is difficult as it combines two challenging requirements: the system must ensure **safety constraints** and it often demands **many resources**. These resources are performance metrics such as tim-

ings, energy consumption and chip temperature.

Currently, the approach used to design modern embedded systems of our daily lives is to specify their features in a "high-level" programming language. In other words, to design a useful feature for the system, a system designer writes code that looks like the following oversimplified example:

```
do {
    useful_work();
}
```

While this approach has many advantages, it has a clear shortcoming: it specifies only what must be the *functional behaviour* of the feature and not its **non-functional requirements** such as *resource usage*.

The real-time systems research community is working towards the creation of techniques, frameworks, policies, tools and languages in order to be able to take into account the **timing behaviour** of computer software. In other word, the very high-level and long term objective of the field would be to replace, in a designer methodology, the former piece of pseudo-code by the following:

```
do in max 30 ms {
    useful_work();
}
```

In the thesis, we complement this approach by adding multiple other criteria into consideration, such as energy consumption and thermal effects. For example:

```
do in max (30 ms, 20 mJ, 2 °C) {
    useful_work();
}
```

This last piece of code would mean to ask the following to the system: *"dear operating system, could you please execute my task (or function) called* `useful_work()` *in less than 30 milliseconds, by consuming less than 20 millijoules and with an increase of the platform temperature of no more than 2 degrees Celsius?"*

Or, in a more classic "human-to-machine" interaction, the following compiler command-line terminal:

```
$ cc useful.c
```

After some processing, and if the selected hardware platform matches these given system constraints, the hypothetical metric-aware compiler would return the following result:

```
$ cc useful.c
OK (i.e. schedulable)
$
```

It means that the compiler found a feasible schedule of the "useful" application matching the given non-functional requirements. Or, in case the constraints were not matched:

```
$ cc useful.c
Error - please allocate 2 cores to function
'useful_work' (not schedulable).
$
```

The compiler would inform the user that it was not able to build (or "schedule") the system due to too demanding constraints. It suggests a solution to the system designer to match the given requirements, namely *allocating more cores* to the application. This suggested solution is based on **exploiting the parallelism** available on the selected hardware platform.

Of course, the results achieved in the research work presented in the dissertation are nowhere close to this ideal vision and long term goal abstractly expressed in the form of C pseudo-code. However, we firmly believe that we took some steps into that direction.

At a very high-level, our contributions can be summarised as follows: we study the techniques to meet non-functional requirements of real-time low-power embedded system design, with a focus on **timing requirements** and **energy consumption**. The techniques are based on exploiting **inherent parallelism** present on modern hardware platforms.

We studied the state of the art in the field and complement it with both *theoretical results* drawn from mathematical models and *practical validations* through measurement-based experiments performed on real-world hardware and software platforms.

In particular, we prove the following claim in both theory and practice: "**using parallelism** allows for a significant **reduction in energy consumption** while still meeting the application **real-time requirements**".

In the context of a university *spin-off* project and for the purpose of the dissertation's practical experiments, we developed and adapted a new multi-core Real-Time Operating System (RTOS) for embedded systems, including a parallel micro-kernel. This RTOS, called HIPPEROS, is part of a solution package aiming to assist system designers for the development of embedded systems under performance and low-power constraints.

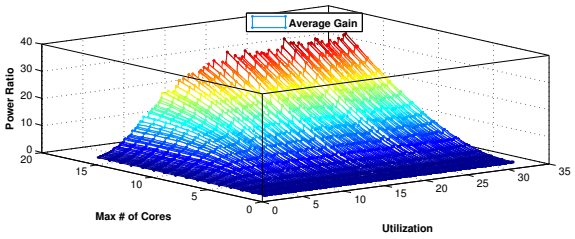
3 Chosen results

Power-aware computing is at the forefront of embedded systems research due to market demands for increased battery life in portable devices and decreasing the carbon footprint of embedded systems in general. The drive to reduce system power consumption has led embedded system designers to increasingly utilise multi-core processing architectures. An oft-repeated benefit of multi-core platforms over computationally-equivalent single-core platforms is increased power efficiency and thermal dissipation. For these power benefits to be fully realised, a computer system must possess the ability to parallelise its computational workload across the multiple processing cores. However, parallel computation often comes at a cost of increasing the total computation that the system must perform due to communication and synchronisation overhead of the cooperating parallel processes.

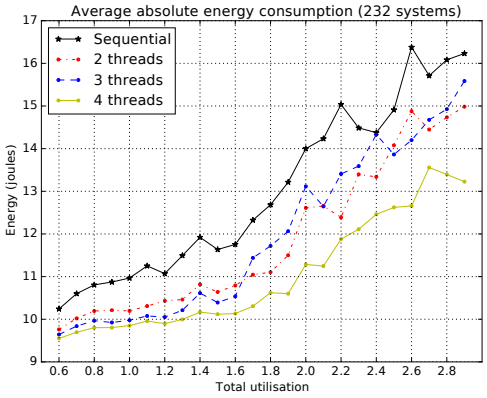
In the thesis, we explored the trade-off between parallelisation of real-time applications and savings in the power and energy consumption. We chose to present two main results in this summary. Each result led to different papers published in international conferences.

3.1 Theoretical benefits of parallelism for power-efficient systems

The first result is drawn for the mathematical model of the problem. We investigated the potential benefit of parallelisation for both meeting real-time constraints and minimising power consumption. Obtaining power efficiency for real-time systems is a non-trivial problem due to the fact that processor power-management features (*e.g.*, clock throttling/gating, dynamic voltage/frequency scaling, etc.) often increase the execution time of jobs and/or introduce switching time overheads in order to reduce system power consumption. The increased execution time for jobs naturally puts additional temporal constraints on a real-time system. Job-level parallelism can potentially help reduce these constraints by distributing the computation to reduce the



(a) Expected power gains of parallel systems over their sequential equivalents in a theoretical framework. The power ratios are computed for varying degrees of parallelism ("Max # of Cores") and varying software workload ("Utilization"). Ratios up to 36 can potentially be obtained when switching from sequential programming to parallel programming.



(b) Practical evaluations of energy consumption of embedded systems. Each curve represents systems with varying software workload ("Total utilisation") but with a fixed degree of parallelism (the number of threads). We can note that increasing the degree of parallelism leads to better energy-efficient systems.

Figure 1 – Illustrations of two main results of the thesis.

elapsed execution time of a parallel job. However, the trade-offs between parallelism, increased communication and synchronisation overheads, and power reduction form a complicated and non-linear relationship. We considered scheduling of parallel real-time tasks upon multiprocessors. We were able to derive an *offline* polynomial-time optimal processor/frequency-selection algorithm. As it can be noted on Figure 1a, simulations of our algorithm on randomly generated task systems with varying workload executing on platforms having up to 16 processing cores show that the theoretical power consumption is reduced by a factor up to 36 compared to the optimal non-parallel approach.

We provided a *theoretical* evaluation of the potential reduction in system power consumption that could be obtained by exploiting parallelism of real-time applications.

3.2 Experimental evaluations of power-aware parallel systems

The second main result of the thesis consists in evaluating parallel implementations of real-time applications upon an *actual* hardware test bed and operating system. It is primarily motivated by the power savings observed in the simulations described above.

We showed with practical experiments that **real-time parallel systems can save more energy than their sequential equivalents** for the same workload. More specifically, we showed that with appropriate mechanisms, the energy savings depend on the *degree of parallelism* of the user tasks.

We proposed a practical, system-level framework to define parallel hard real-time systems and schedule them with a power-aware policy. We provide a realistic experimental evaluation of our framework by implementing a set of simple and easy to parallelise use case programs, mapping them to randomly generated (but schedulable) hard real-time systems and running them on top of a real-time operating system deployed on an embedded system. Power measurements are made with an oscilloscope to evaluate the potential benefit of exploiting parallelism to improve energy efficiency on real running systems. Results of these measurements are illustrated by Figure 1b.

As our results show that **increasing the degree of parallelism** (*i.e.*, adding threads to the execution of a task) **tends to lead to better energy savings**, it suggests **a wider adoption of power-aware intra-task parallelism techniques** for the design of embedded real-time applications.