Modeling and simulation of power consumption and execution times for real-time tasks on embedded heterogeneous architectures

Alessio Balsini, Luigi Pannocchi, Tommaso Cucinotta
Scuola Superiore Sant'Anna
Pisa, Italy
firstname.lastname@santannapisa.it

ABSTRACT

In this work, we introduce a power-consumption model for heterogeneous multicore architectures that captures the variability of energy consumption based on processing workload type, in addition to the classical variables considered in the literature, like type and frequency of the CPU.

We motivate the approach presenting experimental results gathered on a Odroid-XU3 board equipped with an Arm big.LITTLE CPU, showing that power consumption has a non-negligible dependency on the workload type. We also present a model to define the execution time of the tasks, which depends on both the workload, and the CPU frequency and architecture. We present our modifications to the open-source RTSIM real-time scheduling simulator to extend its CPU power consumption and execution time duration models, integrating results taken from the real platform.

The presented work constitutes a useful base for future research in power-aware real-time scheduling on heterogeneous platforms.

CCS CONCEPTS

• **Computer systems organization → Real-time operating systems**; **Embedded systems**;

KEYWORDS

Power efficiency, simulation, heterogeneous architectures, real-time systems, power-aware execution-time modeling

ACM Reference Format:
Alessio Balsini, Luigi Pannocchi, Tommaso Cucinotta. 1997. Modeling and simulation of power consumption and execution times for real-time tasks on embedded heterogeneous architectures. In *Proceedings of Embedded Operating System Workshop (EWILI 2018).* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Mobile computing is flourishing as an essential tool to support our daily activities, with relatively powerful battery-operated and interconnected devices, capable of hosting an operating system (OS) and a plethora of interactive applications. These devices have evolved in hardware capabilities over the last few years, with a growing number of connectivity options, an unimaginable growth rate for their volatile and persistent storage sizes and increasing computational power, in the form of multicore architectures. In this context, energy management has been receiving a lot of attention from both research [15, 16] and industrial communities, as energy efficiency is now one of the top concerns when designing new devices, functionality, applications and services. These motivations led to the development of new platforms that are focused on heterogeneous processors with a unified ISA, such as the Arm big.LITTLE$^{(RM)}$. These, departing from traditional symmetric multi-processing (SMP) architectures, possess both low-complexity, low-power cores specializing in device bookeeping activities, and high-power cores for CPU-intensive activities, in addition to classical frequency switching capabilities, where tasks can be migrated among all of the cores as needed.

This is causing the urgent need for engineering new functionality at the OS level, and specifically regarding joint CPU scheduling, task placement and energy management, where proper and novel trade-offs among energy consumption and interactivity of devices and applications with the outside world have to be sought. This is witnessed, for example, by the recent volume of activities around the Energy-Aware Scheduling (EAS) framework in the Linux kernel[1], engineered around the support for Arm-based CPUs in Android.

To this end, it is essential to rely on accurate models of the underlying hardware behavior and the impact of the available energy-management tunables on the application performance. These models can be embedded within proper simulation tools that allow for estimating the expected impact of novel energy management and task scheduling features at the OS level on the final application performance and its capability to respect possible timing constraints.

1.1 Problem Presentation and Contributions

Aiming at reproducing the energy behavior of computational activities within embedded heterogeneous architectures, it is necessary to take into account the power consumption of the CPUs and the time necessary to complete the computations, i.e., the execution times. These metrics allow to calculate the energy involved in the computation process and investigate on interesting trade-offs between computation performance and energy efficiency.

In this work, we propose a support to tackle the problem of power-aware CPU scheduling in real-time systems based on heterogeneous, single-ISA architectures by adopting a *workload-dependent* power consumption model, coupled with a *workload-dependent*

[1] More information is available at https://lwn.net/Articles/749738/ and https://developer.arm.com/open-source/energy-aware-scheduling.

EWILI 2018, 2018, Turin, Italy

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

execution-time scaling model, at the varying of CPU frequencies. We base our proposal on experimental results highlighting shortcomings of the commonly adopted simple scaling models at varying CPU frequencies, assuming power consumption as a quadratic function of the CPU frequency [26]. Indeed, in Figure 1 we report the measured power consumption of two real, different workloads on our reference big.LITTLE board (Odroid-XU3), highlighting that the power consumption model also strongly depends on the workload type. Our measurements show, for example, that at maximum frequency an application performing data encryption (`encrypt`) has a power consumption that is 38% higher than a memory-bound application generating a huge number of cache misses (`cachekiller`), forcing the CPU into continuous stalls waiting for completion of memory accesses. Real workloads pose themselves somewhere in the middle between these two extremes, depending on the instruction mix and data access pattern. Moreover, the same figure shows that the also the power consumption of the CPUs in a clock-gating idle state depends on the frequency.

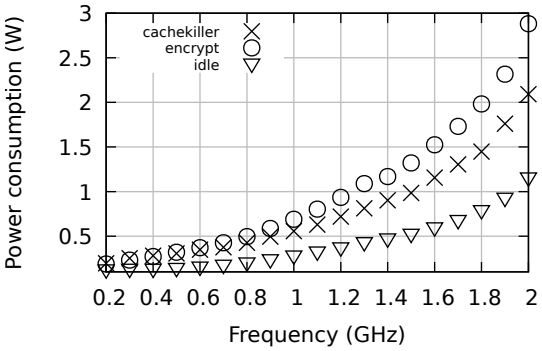

Figure 1: Measurements on the power consumption of the big CPU cluster, running different workloads at different frequencies.

For non-continuously running activities, the energy consumption depends also on the duration of the computations, which is affected by the frequency of the CPU in a way that is workload-dependent as well. Classical models rely on a simple scaling of the execution time with the operating frequency of the CPU. However, our experimental results show that the workload type affects this relationship in a non-negligible way. These are compared in Figure 2, where the measured execution time variation for three distinct workload types is reported, normalized with respect to the execution time at the lowest frequency. As evident, some workload types have a significant deviation from the simple scaling model commonly adopted in literature (continuous line), corresponding to Equation (8) in Section 3. More details will follow in Section 5.

The contribution of this work is (i) the development of power consumption and execution time models derived starting from established solutions already available in literature, (ii) their implementation through the open-source *RTSIM* real-time systems simulator, and (iii) the validation of the simulation outcome with the measurements performed on a real platform. This tool constitutes then a valuable means for the preliminary evaluation and testing of novel energy-aware task scheduling algorithms.

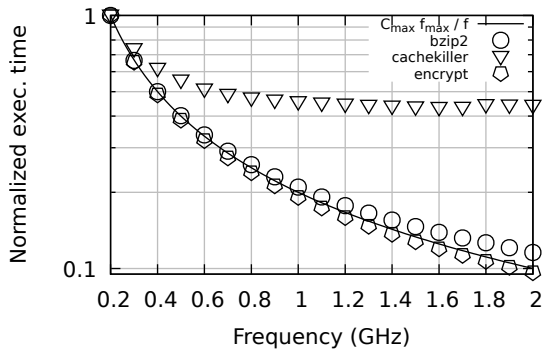

Figure 2: Normalized execution times of different workloads running on a big CPU at different frequencies.

2 RELATED WORK

The research literature related to the present work falls mainly within the classes of power consumption models for voltage-scalable embedded and heterogeneous architectures and simulation of real-time systems including power-aware resource management logic.

Given the growing interest in energy efficient devices, the research communities have already developed several power consumption models for CPUs. The level of details and complexity is variable and depends on the specific application.

As long as the power consumption models are concerned, there are works based on a detailed description of the CPU architecture, such as the one by Möbius et. al. [19], focusing on a CPU model for on-line power estimation using *performance monitoring counter* of the physical CPU. Other detailed models exist, focusing on the electronics behind the CPU operation, like the one developed for the *Wattch* simulator by Brooks et. al. [6]. Indeed, in this case, the goal was the architecture-level power analysis, evaluating the power consumption at the instruction level.

These approaches are very accurate, at the expense of their usability: they require a detailed description of the hardware, and are often characterized by long computation times for the simulation. In this paper, we adopt a higher-level abstraction approach, which uses real data carried out over a set of micro-benchmarks. This allows for a realistic reproduction of the CPU power consumption pattern throughout an application execution, without the need for considering specific architectural details. This way, we can achieve a trade-off between simulation efficiency and representativeness.

In that direction, high-level descriptions of the CPU power are preferred, like the ones leveraging on more coarse physical models. The interest in these models is justified by the fact that the power consumption of the CPU is modeled using physical quantities like frequency, currents and voltages, which are easier to access. This approach is proposed by several works [4, 7, 17, 25], describing the behavior of real platforms, relying on simplified models with a limited number of parameters and input variables. As demonstrated by Colin et. al. [10], the fitting of a non-linear model with only six parameters over the power behavior of a real board provides sufficient accuracy.

Our work uses a more complete power consumption model, obtained taking the cue from the physical behavior of the CPU, close to the one used by Vogeleer et. al. [12]. The parameters of the model have been identified through real experiments, which constitute also a validation process for the final system.

In real-time systems research literature, significant investigations have been carried out in energy-consumption models and energy-aware task scheduling algorithms guaranteeing system schedulability or minimum levels of quality of service (QoS) while at the same time trying to realize energy-efficient policies on hardware supporting dynamic voltage and frequency scaling (DVFS). In this field, a reliable model for the execution time is of paramount importance, both for providing an accurate estimation of the power consumption, and for guaranteeing timing requirements for latency-sensitive applications. Often the computation time models are based on strict theoretical assumption and simplifications, which are far from the operating condition of a real system. Other works, on the contrary, go more into details and model the execution time or delays considering low level hardware information for the given platform. Works like the one by Palacharla et. al. [20] analyze the hardware details of the architecture to quantify the computational speed, but they are too complex and not practical. A more practical approach, which avoids considering hardware details, is shown in the work of Petrucci et. al. [23], where the computation time of a task depends on the accesses to the memory and number of instructions to be executed by the CPU. Despite the realism achieved thanks to the empirical identification of the model parameters, this work only considers CPUs with fixed frequency. Vogeleer et. al. [12] model instead the computation time of the task as a simple function of the frequency, thus allowing considering the effects of CPU frequency scaling.

Compared to the latter approach, ours is still a simple yet representative model, thanks to additional parameters. This achieved acceptable accuracy in our preliminary experimentation.

Other works related to power consumption modeling and scheduling optimization exist in the domain of high-performance computing and many-core systems, but their review is omitted.

Concerning the existence of complete frameworks able to reproduce the overall behavior of a full platform, several tools exist for modeling, simulation and schedulability analysis of embedded, real-time distributed systems, including *RTSIM* [3, 21], *MAST2* [14], *TIMES* [2], *SimTrOS* [5], *YARTISS* [8], *FORTAS* [11], *McSimA+* [1] and others. However, the main purpose of these is to tackle the task-set schedulability problem, not considering the energy impact.

The *SimDVS* [24] by Shin et. al., instead, addresses the problem, evaluating the impact of the scheduling decisions on the energy consumption, but is limited to single processor architectures.

Considering the above, it is our belief that a simulation framework supporting real-time scheduling and a more precise energy model for modern heterogeneous multicore architectures is a useful tool for future research in the area.

3 PROPOSED APPROACH

3.1 Model of the Power Consumption

As widely known [7, 17], the power consumption of a CPU is determined by different phenomena, related to its transistor nature: switching activity and leakage effects.

Switching Activity. The switching of transistors during the computation requires power to charge the gates capacitors; moreover, it gives birth to some power loss due to brief short-circuit conditions during toggling among logic levels. In the following, the former is called P_{cg}, the latter P_{sc}. As done by Vogeleer et. al. [12], the equations describing these quantities are the following:

$$P_{cg} = \alpha C V^2 f \quad \text{and} \tag{1}$$

$$P_{sc} = \eta P_{cg}, \tag{2}$$

where C is the capacitance of the transistor gates, f is the CPU switching frequency and V is the CPU voltage. P_{sc} has been considered to be proportional to P_{cg} by a factor η. This choice simplifies the model, without compromising the representativeness. The α factor represents the number of transistors involved in the switching, that is the activity level of the CPU. In our proposed model, α is the quantity that is mostly related to the workload type being run by the CPU, ultimately leading to different possible power consumption levels. Precisely, α is assumed to vary in a range $[\alpha_0, \alpha_{max}]$, where the lower bound α_0 is associated with the idle state of the CPU, whilst the upper bound α_{max} is associated with the CPU under an intensive workload. The overall power consumption due to the switching activity P_{sw} is given by the sum of the two, that is,

$$P_{sw} = P_{cg} + P_{sc}. \tag{3}$$

Leakage Effects. Despite the technological improvements in the semiconductor device fabrication, there are always some leakage currents flowing between different parts of the transistor. This effect determines a power loss that will be referred to as P_{lk}. The definition of a complete model to represent this phenomena is complex and depends on several variables, including the temperature [18]. To overcome this complexity, some simulators like *Wattch* model this component as a fixed percentage of the dynamic power, others include a simple thermal modeling to improve the accuracy, like TEM^2P^2EST [13]. From the work of Skadron et al. [25], it turns out that there is a relationship between the leakage and the switching powers. More precisely,

$$P_{lk} = \left(\frac{R_0}{V_0 T_0^2} e^{\left(\frac{B}{T_0} + \frac{B}{T}\right)} T^2 V\right) P_{sw}, \tag{4}$$

where T_0 is the ambient temperature, B is a constant, V_0 is the nominal voltage and R_0 the ratio between P_{lk} and P_{sw} when $T = T_0$. It is worth highlighting that the equation is temperature dependent. However, in the case of stable temperature, part of the expression can be approximated by a constant factor. This leads to $P_{lk} = \gamma V P_{sw}$, where

$$\gamma = \frac{R_0}{V_0 T_0^2} e^{\left(\frac{B}{T_0} + \frac{B}{T}\right)} T^2. \tag{5}$$

Summarizing, the total power consumption of the CPU, when working at frequency f, voltage V and stable temperature, is:

$$P_{CPU} = P_{sw} + P_{lk} = (1 + \eta)(1 + \gamma V)\alpha C f V^2. \tag{6}$$

In this work, the constants of the model depend on the operating conditions, i.e., during the *idle* state of the CPU, those parameters are expected to be different from the ones of the model representing the CPU performing computational activities. Moreover, each kind of workload has a specific usage of the CPU, inducing a different power demand: these considerations are taken into account when identifying the parameters of the power consumption model.

3.1.1 Power Model Identification. As stated above, the model used in this simulation framework is a trade-off between complexity and representativeness, leading to a "gray box" model. Therefore, the model parameters have been identified by fitting the outcome of experiments running a set of micro-benchmarks on a real platform.

The identification procedure has been accomplished using NeuroLab[2], a tool for data fitting based on genetic algorithms. Specifically, the function used for the fitting is

$$P_{CPU} = \delta + (1 + \eta)(1 + \gamma V)KfV^2, \tag{7}$$

where the αC of the Equation 6 has been substituted with a single parameter K, which, together with δ, η and γ, constitute the parameters of the fitting model. The additional parameter δ introduces a further degree of freedom in the function fitting. Given the dependence of the power consumption on the workload, the fitting has been accomplished for each workload type in a given set, which will be described in Section 5. The γ parameter has been considered constant, assuming the temperature as stable during each benchmark. Even though this could be a limitation, a proper modeling would require considering also the thermal model of the CPU, which is left to future works.

3.2 Workload-dependent Execution Time Model

The execution time C of a task running on a CPU with DVFS capabilities is typically assumed as a simple function of the frequency:

$$C(f) = C_{max}\frac{f_{max}}{f}, \tag{8}$$

where C_{max} denotes the execution time at the maximum frequency f_{max} on the same CPU. This is based on the assumption that the time required by a task to complete only depends on the frequency. In reality, as introduced in Section 1, there are other factors not considered in this model, affecting the task duration. For example, the memory access time does not scale with the frequency, thus represents a bottleneck partially mitigated by the use of caches.

An analytical model we found out to be able to reproduce the behavior of real workloads at the varying of the CPU frequency with a reasonable number of free parameters a, b, c, d is the following:

$$C(f) = a + \frac{b}{f} + ce^{-f/d}, \tag{9}$$

which includes, in order: (i) a fixed offset that models the presence of bottlenecks for which the speed does not depend on the CPU frequency, (ii) an hyperbolic component that models the ideal execution time scaling with the frequency, and (iii) an adjustment on the function slope.

[2]More information at: https://github.com/balsini/NeuroLab.

To achieve the maximum accuracy, the execution time model should also consider, amongst all, the interference introduced by the other tasks running in the system and causing cache and bus contention, and hardware devices accessing the bus with DMA operations. This level of detail is beyond the purpose of this work, for which we provide an execution time model that applies for single tasks running in the system, which still represents a valid approximation for a number of applications.

4 IMPLEMENTATION DETAILS

The power consumption and execution time models presented above have been implemented within the *RTSIM* simulator. This is a portable, extensible open-source package written in C++ for the simulation of real-time operating systems, supporting many real-time scheduling policies and typical real-time task models. *RTSIM* carries out a high-level simulation focusing on the timing and schedule of tasks in the system, without any functional-level simulation. Its typical use is to simulate worst-case scheduling scenarios for real-time task sets under a given scheduling policy, for the purpose of verifying whether any deadline miss happened or not. *RTSIM* includes a library for discrete event simulation (*METASIM*), and a set of libraries for real-time kernels simulation (*RTLib*). It also provides functionality to trace, store and visualize the events occurred in the simulated environment.

The simulation of a multicore computing system in *RTSIM* involves several modules, among which the most important are:

- `Task`: entity that executes for a given amount of time. A task can also be periodic, as common for real-time environments.
- `Scheduler`: the scheduling policy for the tasks running in the system, for example EDF, global EDF, fixed priority FIFO or Round-Robin, and others.
- `CPU`: modulates the duration of the task with the ideal linear model shown in Equation 8, and provides the basic power consumption estimation as $P(f) = V^2 f$.
- `Kernel`: a glue entity that connects the tasks with the scheduler and the CPUs, and manages possible virtual resources shared among tasks, like semaphores.
- `Trace`: module to store events and accumulate statistics.

Once the system is initialized, the simulation runs and evolves with the simulated entities generating, exchanging and executing simulation events.

Our extensions to *RTSIM*[3] include improvements on the `CPU`, `Task` and `Trace` classes, and the implementation of the presented power consumption and execution time models with the `CPUModel` class. With our modifications, when a Task is put in execution, it can now declare the type of workload the task is going to execute, through an extension to the `fixed()` virtual instruction type of the task. This instruction now also specifies the workload type, in addition to the already available execution time of the computation, calibrated at the maximum frequency supported on the highest performance CPU in the system. *RTSIM* automatically adjusts the computation duration and the consumed power when scheduling that task executing each virtual instruction, according to the provided `CPUModel` parameters supplied at CPU instantiation time, the CPU capabilities, the frequency and the workload type.

[3]Freely available at: https://github.com/balsini/rtlib2.0/tree/ewili-2018.

The `TracePowerConsumption` class implements a power measurement probe, and it is possible to create one instance for each CPU, tracking the power consumption by querying the power model of the associated CPU.

5 EXPERIMENTAL RESULTS

The simulator has been tested by comparing its simulated results with the ones obtained by experiments on the real platform.

All the experiments presented in this section refer to an Odroid-XU3 board, which embeds a Samsung Exynos 5422 SoC: an Arm big.LITTLE architecture with four Cortex-A7 and four Cortex-A15 CPUs, running the official Odroid Linux kernel 3.10. Similar results have been achieved with a Linux kernel 4.15. All the power measurements have been performed with the INA231 power meters already available within the board.

5.1 Experiments to Gather Model Fitting Data

In order to collect detailed data sufficient to fit our workload-dependent power consumption model, a number of different workload types have been experimented with: (i) `idle`: no task is running, so the system switches to the clock-gating idle state [22]. Our board supports two idle states, and the deeper can be accessed only when all the CPUs are idle, thus is never considered in our experiments. (ii) `bzip2`[4]: compression algorithm, with maximum compression level. (iii) `des3 encrypt/decrypt`[5]: Triple DES encryption algorithm. (iv) `sha256sum`[6]: checksum algorithm. (v) `cachekiller`: application written with the purpose of generating a cache miss at every iteration, by accessing elements within an array bigger than the cache memory size, every access performed with a displacement bigger than the cache line.

All the data read or written by the aforementioned data intensive workloads is randomly generated, and stored in a *ramfs* mounted partition to avoid possible latencies or throughput limitations due disk or SD card devices.

To characterize the system behavior for the two different CPUs, each experiment is run by sequentially pinning the workload task on one of the big cores first, and on one of the LITTLE cores later.

Each experiment on the real platform is repeated 50 times and each final data point is obtained as an average on the measured values, which have been found to have a small variability from run to run. The observed standard deviation of the measured values for each experiment, normalized with respect to the average among the values, has consistently been in the range between 4% and 12.1%.

5.2 Experiments With our Modified RTSIM

In a first experiment, we evaluate the power consumption model implemented in the simulator. As shown in Figure 3, the simulated behavior successfully maps on the experimental results for the presented workloads. On the other hand, the LITTLE CPU presents a noticeable error in the mid-range frequencies. This effect is likely due to the CPU power supplier, which may be composed of multiple circuits [9] causing a discontinuous behavior.

[4]Bzip2 1.0.6, available at: http://www.bzip.org.
[5]OpenSSL 1.0.2g, available at: https://www.openssl.org.
[6]GNU coreutils 8.25, available at: http://www.gnu.org/software/coreutils/sha256sum.

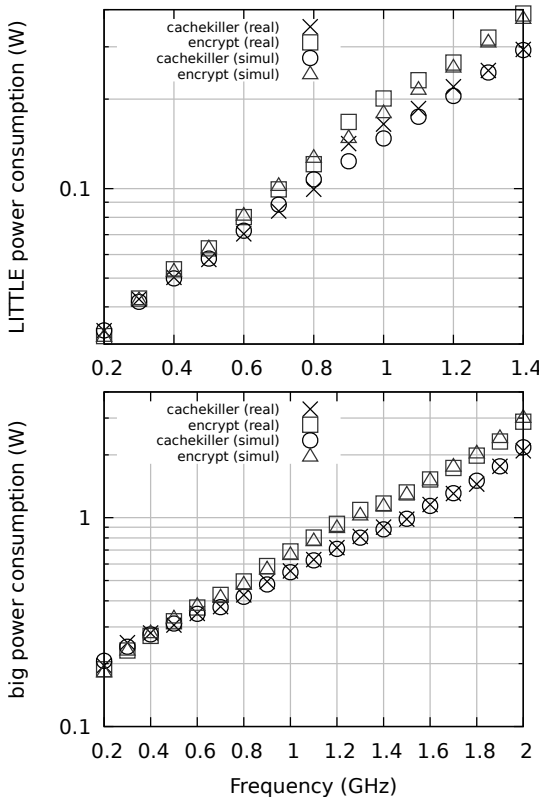

Figure 3: Comparison between the power consumption simulated with *RTSIM* and the respective experimental results.

For simplicity, the plot only shows a subset of the tested workloads. Among all the workloads, the highest relative least square error measured in the comparison between the experimental and simulated results is associated to the `cachekiller` running on a LITTLE CPU, and has a value of 16.1%.

The next experiment evaluates the execution time model for the example workloads running on different CPUs at different frequencies. In this case, as demonstrated in Figure 4, the model behavior is definitely close to the experimental data on the real platform.

As in the previous experiment, the highest relative least square error measured in the comparison between the experimental and simulated results is associated to the `cachekiller` running on a LITTLE CPU, and has a value of 3.57%.

6 CONCLUSIONS

This paper presented an effective modeling approach to reproduce the energy and timing behavior of a heterogeneous multicore architecture, running real-time tasks and under different workload conditions. The work dealt also with the implementation of the proposed models on the *RTSIM* simulator, extending its capabilities to obtain a comprehensive suite for simulation of energy-aware strategies.

The extended *RTSIM* real-time scheduling simulator has been tested through experiments, checking the output of the simulated scenarios with respect to the real cases. The accuracy of the results

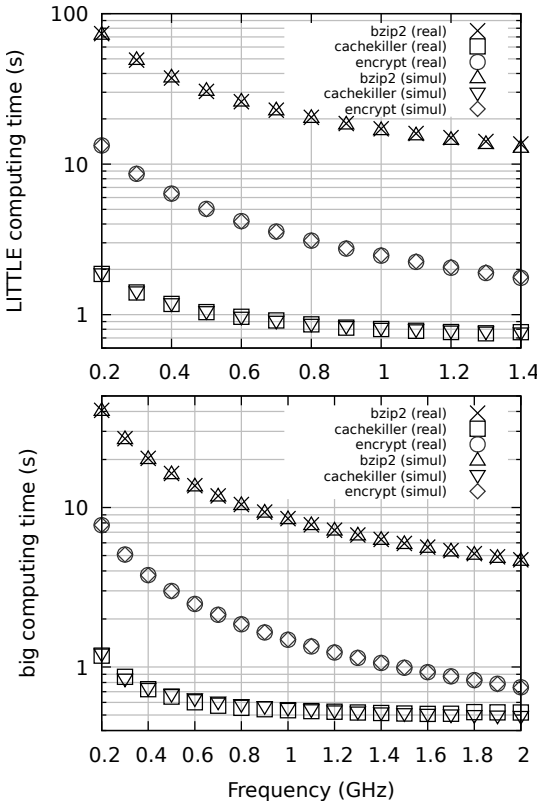

Figure 4: Validation of the execution times model of the simulator by comparing the experimental results.

in terms of simulated execution times and energy consumption showed that this work represents a valuable tool for the evaluation and testing of novel energy-aware task scheduling algorithms.

As a future work, we consider the extension of the simulation environment to improve the modeling of the power consumption and execution time for complex workload patterns in terms of heterogeneity of the workload and degree of execution parallelism. On the other hand, it would be useful to investigate further modeling approaches for describing generic workloads, i.e., memory bound or CPU bound.

REFERENCES

[1] J. H. Ahn, S. Li, S. O, and N. P. Jouppi. 2013. McSimA+: A manycore simulator with application-level+ simulation and detailed microarchitecture modeling. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 74–85. https://doi.org/10.1109/ISPASS.2013.6557148

[2] Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. 2004. TIMES: A Tool for Schedulability Analysis and Code Generation of Real-Time Systems. In *Formal Modeling and Analysis of Timed Systems*, Kim Guldstrand Larsen and Peter Niebert (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 60–72.

[3] Cesare Bartolini and Giuseppe Lipari. 2011. The RTSIM scheduling simulator. (2011). http://rtsim.sssup.it/

[4] R. Basmadjian and H. de Meer. 2012. Evaluating and modeling power consumption of multi-core processors. In *2012 Third International Conference on Future Systems: Where Energy, Computing and Communication Meet (e-Energy)*. 1–10. https://doi.org/10.1145/2208828.2208840

[5] Michael Bohn, Jörn Schneider, and Christian Eltges. 2013. SimTrOS: A heterogenous abstraction level simulator for multicore synchronization in real-time systems. *Journal of Systems Architecture* 59, 6 (2013), 297 – 306. https://doi.org/10.1016/j.sysarc.2013.02.002

[6] D. Brooks, V. Tiwari, and M. Martonosi. 2000. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*. 83–94.

[7] J. A. Butts and G. S. Sohi. 2000. A static power model for architects. In *Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000*. 191–201. https://doi.org/10.1109/MICRO.2000.898070

[8] Younès Chandarli, Frédéric Fauberteau, Damien Masson, Serge Midonnet, and Manar Qamhieh. 2012. YARTISS: A Tool to Visualize, Test, Compare and Evaluate Real-Time Scheduling Algorithms. In *WATERS 2012*. Italy, 21–26. https://hal-upec-upem.archives-ouvertes.fr/hal-00691985

[9] W. H. Cheng and B. M. Baas. 2008. Dynamic voltage and frequency scaling circuits with two supply voltages. In *2008 IEEE International Symposium on Circuits and Systems*. 1236–1239. https://doi.org/10.1109/ISCAS.2008.4541648

[10] A. Colin, A. Kandhalu, and R. Rajkumar. 2014. Energy-efficient allocation of real-time applications onto Heterogeneous Processors. In *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*. 1–10. https://doi.org/10.1109/RTCSA.2014.6910506

[11] P. Courbin and L. George. 2011. Fortas: Framework for real-time analysis and simulation. In *Proc. of WATERS 2011*. Porto, Portugal, 21–26.

[12] Karel De Vogeleer, Gerard Memmi, Pierre Jouvelot, and Fabien Coelho. 2014. The Energy/Frequency Convexity Rule: Modeling and Experimental Validation on Mobile Devices. In *Parallel Processing and Applied Mathematics*, Roman Wyrzykowski, Jack Dongarra, Konrad Karczewski, and Jerzy Waśniewski (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 793–803.

[13] Ashutosh Dhodapkar, Chee How Lim, George Cai, and W. Robert Daasch. 2001. TEM2P2EST: A Thermal Enabled Multi-model Power/Performance ESTimator. In *Power-Aware Computer Systems*, Babak Falsafi and T. N. Vijaykumar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 112–125.

[14] Michael González Harbour, J. Javier Gutiérrez, José M. Drake, Patricia López Martínez, and J. Carlos Palencia. 2013. Modeling distributed real-time systems with MAST 2. *Journal of Systems Architecture* 59, 6 (2013), 331 – 340. https://doi.org/10.1016/j.sysarc.2012.02.001

[15] Connor Imes and Henry Hoffmann. 2015. Minimizing Energy Under Performance Constraints on Embedded Platforms: Resource Allocation Heuristics for Homogeneous and single-ISA Heterogeneous Multi-cores. *SIGBED Rev.* 11, 4 (Jan. 2015), 49–54. https://doi.org/10.1145/2724942.2724950

[16] C. Imes, D. H. K. Kim, M. Maggio, and H. Hoffmann. 2015. POET: a portable approach to minimizing energy under soft real-time constraints. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. 75–86. https://doi.org/10.1109/RTAS.2015.7108419

[17] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan. 2003. Leakage current: Moore's law meets static power. *Computer* 36, 12 (Dec 2003), 68–75. https://doi.org/10.1109/MC.2003.1250885

[18] Weiping Liao, Lei He, and K. M. Lepak. 2005. Temperature and supply Voltage aware performance and power modeling at microarchitecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 7 (July 2005), 1042–1053. https://doi.org/10.1109/TCAD.2005.850860

[19] C. Möbius, W. Dargie, and A. Schill. 2014. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (June 2014), 1600–1614.

[20] S. Palacharla, N. P. Jouppi, and J. E. Smith. 1997. Complexity-Effective Superscalar Processors. In *Conference Proceedings. The 24th Annual International Symposium on Computer Architecture*. 206–218. https://doi.org/10.1145/264107.264201

[21] Luigi Palopoli, Giuseppe Lipari, Gerardo Lamastra, Luca Abeni, Gabriele Bolognini, and Paolo Ancilotti. 2002. An Object-oriented Tool for Simulating Distributed Real-time Control Systems. *Softw. Pract. Exper.* 32, 9 (July 2002), 907–932. https://doi.org/10.1002/spe.467

[22] Preeti Ranjan Panda, B. V. N. Silpa, Aviral Shrivastava, and Krishnaiah Gummidipudi. 2010. *Power-efficient System Design* (1st ed.). Springer Publishing Company, Incorporated.

[23] Vinicius Petrucci, Orlando Loques, Daniel Mossé, Rami Melhem, Neven Abou Gazala, and Sameh Gobriel. 2015. Energy-Efficient Thread Assignment Optimization for Heterogeneous Multicore Systems. *ACM Trans. Embed. Comput. Syst.* 14, 1, Article 15 (Jan. 2015), 26 pages. https://doi.org/10.1145/2566618

[24] Dongkun Shin, Woonseok Kim, Jaekwon Jeon, Jihong Kim, and Sang Lyul Min. 2003. SimDVS: An Integrated Simulation Environment for Performance Evaluation of Dynamic Voltage Scaling Algorithms. In *Power-Aware Computer Systems*, Babak Falsafi and T. N. Vijaykumar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 141–156.

[25] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, Wei Huang, Sivakumar Velusamy, and David Tarjan. 2004. Temperature-aware Microarchitecture: Modeling and Implementation. *ACM Trans. Archit. Code Optim.* 1, 1 (March 2004), 94–125. https://doi.org/10.1145/980152.980157

[26] X. Zhong and C. Xu. 2007. Energy-Aware Modeling and Scheduling for Dynamic Voltage Scaling with Statistical Real-Time Guarantee. *IEEE Trans. Comput.* 56, 3 (March 2007), 358–372. https://doi.org/10.1109/TC.2007.48