Worst-Case Energy-Consumption Analysis by Microarchitecture-Aware Timing Analysis for Device-Driven Cyber-Physical Systems

Phillip Raffeck
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Christian Eichler
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Peter Wägemann
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Wolfgang Schröder-Preikschat
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Abstract

Many energy-constrained cyber-physical systems require both timeliness and the execution of tasks within given energy budgets. That is, besides knowledge on worst-case execution time (WCET), the worst-case energy consumption (WCEC) of operations is essential. Unfortunately, WCET analysis approaches are not directly applicable for deriving WCEC bounds in device-driven cyber-physical systems: For example, a single memory operation can lead to a significant power-consumption increase when thereby switching on a device (e.g., transceiver, actuator) in the embedded system.

However, as we demonstrate in this paper, existing approaches from microarchitecture-aware timing analysis (i.e., considering cache and pipeline effects) are beneficial for determining WCEC bounds: We extended our framework on whole-system analysis with microarchitecture-aware timing modeling to precisely account for the execution time that devices are kept (in)active. Our evaluations based on a benchmark generator, which is able to output benchmarks with known baselines (i.e., actual WCET and actual WCEC), and an ARM Cortex-M4 platform validate that the approach significantly reduces analysis pessimism in whole-system WCEC analyses.

2012 ACM Subject Classification Hardware → Static timing analysis; Hardware → Power and energy; Computer systems organization → Embedded and cyber-physical systems

Keywords and phrases WCEC, WCRE, WCET, michroarchitecture analysis, whole-system analysis

Digital Object Identifier 10.4230/OASIcs.WCET.2019.4

Supplement Material Source code: `https://gitlab.cs.fau.de/syswcec-uarch`

Funding This work is supported by the German Research Foundation (DFG), in part by Research Grant no. SCHR 603/9-2, no. SCHR 603/13-1, no. SCHR 603/14-2, and the Bavarian Ministry of State for Economics, Traffic and Technology under the (EU EFRE funds) grant no. 0704/883 25.

Acknowledgements We thank Simon Schuster for his insightful comments and the support with Platin's source base. We also thank Dominik Huber, Aaron Strahlberger, and Julius Wiedmann for their help with reducing the pessimism of the microarchitecture analysis.

1 Introduction

Most embedded systems have restrictions on their execution performance and energy availability. Tasks in such systems must, therefore, execute within execution-time and energy-consumption budgets arising from the system's schedule and the state of charge. To guarantee the execution within these budgets, reliable upper bounds on the worst-case execution time (WCET) and the worst-case energy consumption (WCEC) are essential. In contrast to WCEC analysis, numerous WCET analysis approaches are available [1, 3, 17, 18, 24, 25, 30].

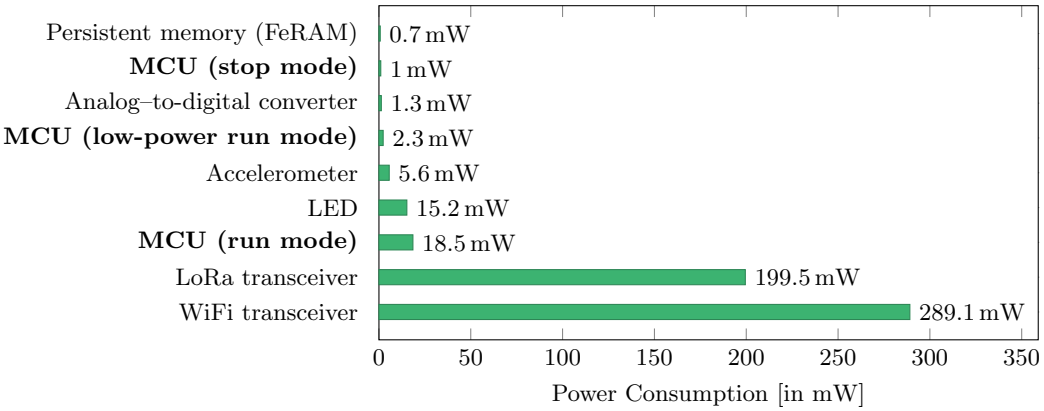


Figure 1 Consumers and their power consumption in systems with devices [7, 13, 15, 37].

In analogy to real-time systems where timely execution must be guaranteed, energy-constrained systems can have hard energy constraints [10], and WCEC bounds on the energy consumption of each task, which is the integral of a task's power consumption over its execution time, are essential to ensure the execution within energy budgets [39, 42]. More specifically, the energy demand from the task's start until its completion with all interferences (i.e., asynchronous interrupts, preemptions) is necessary, which is referred to as the *worst-case response energy (WCRE)*. For example, the WCRE is essential to guarantee the completion of a task storing critical system-state data to persistent memory without risking the battery's depletion during its execution.

These energy-constrained cyber-physical systems are integrated into physical processes, and thus have to interact with their environment, which is achieved by sensors (e.g., analog-to-digital converters), actuators (e.g., motors), or transceivers for communication (e.g., WiFi, LoRa). Figure 1 illustrates the power consumed by devices often integrated into cyber-physical systems. Especially transceivers, being software-controlled by the processor, consume an order of magnitude more than the processor's power demand. Also, CPU-internal (timer) devices significantly contribute to the overall consumption. That is, to precisely determine WCRE bounds, static worst-case analyses have to model these components. Existing approaches to determine WCEC bounds focus on the processor and microarchitecture-aware power modeling [21, 26, 29, 36, 38]. However, they ignore the influence of power-hungry devices.

Considering the temporal behavior of the microarchitecture, including hardware features such as caching and pipelining, is crucial and well explored in WCET analysis to determine sound and precise bounds. For some processors, this temporal behavior is provided by the documentation [20]. Regarding energy consumption, the energetic microarchitectural behavior is not available for commercial platforms. However, the maximum power consumption in specific states is available for some processors [14, 34] and peripherals [7, 37], which is information we use together with the worst-case execution time to bound energy consumptions.

In this paper, we present an approach to precisely account for all context-sensitive power consumers in an embedded system by a microarchitecture-aware WCET analysis. The whole-system WCRE analysis decomposes the input system into blocks of constant maximum power consumption. Based on this abstraction, a system-wide path analysis is conducted that includes all possible preemptions and the fixed-priority scheduling scheme. A microarchitecture-aware timing analysis accounts for the durations the devices are kept (in)active. Finally, with the gathered information, an optimization problem is formulated, whose solution eventually yields WCRE bounds for tasks in the system.

This work is based on prior work on WCRE analysis [40] and extends it by the following aspects: We now consider microarchitectural effects in the processor's pipeline, the influence of caches, and especially cache-related preemption delays in the fixed-priority scheduling scheme. Additionally, we further discuss the necessity to model energy consumption by microarchitecture-aware timing analysis in device-driven embedded systems. Finally, we evaluate our prototype with GenEE [12], which is able to generate device-aware benchmarks with known WCEC paths, which again serve as baselines for the presented evaluations.

The paper is structured as follows: Section 2 outlines the major challenges when determining precise upper bounds on the system's energy consumption. In Section 3, we present our approach to solve these challenges and give an overview of the implementation of our prototype, which is evaluated in Section 4. Section 5 outlines related and future work in the context of system-wide WCEC analyses, and Section 6 concludes.

2 Problem Statement

We aim to tighten the results of whole-system resource analysis by considering the microarchitecture state in the WCRE analysis. In this section, we present an overview of the assumed system and hardware model (see Section 2.1) and identify three challenges that arise due to the incorporation of the microarchitecture state (see Section 2.2–2.4).

2.1 System Model

For the analysis, we assume an OSEK-compliant [28] real-time system comprising tasks, interrupts, and temporarily active devices. According to the OSEK standard, our system model includes a fixed-priority scheduling scheme. Tasks may be interrupted by asynchronous interrupts and, according to their priority, preempted by higher-priority tasks. Furthermore, the task set is known at compile time and configured using OSEK's implementation language (i.e., OIL [27]). Our system model assumes that interrupt service routines (ISRs) have run-to-completion semantics. If ISRs activate higher-priority tasks, these are executed after the handler. All interrupts that potentially preempt a task during execution are known at compile time, and their occurrence is bounded by their minimum inter-arrival time.

All peripheral devices have two different power consumption modes (i.e., on/off), which are software-controlled via explicit system calls. Currently, we do not consider devices that have hardware-induced mode changes or have multiple modes, which are aspects of future work. Switching on a device subsequently leads to a constant increase in power consumption.

We consider small, embedded single-core processors with instruction caches and pipelines, whose complexity is similar to CPUs commonly used in safety-critical, hard real-time systems. An example for our system model is the Infineon XMC4500 [20] – an ARM Cortex-M4 processor we used in our implementation – that features an in-order 3-stage pipeline and a 4 KiB instruction cache (2-way set associative). The instruction cache employs an LRU cache-replacement policy, which is known to be free of timing anomalies [6, 16].

2.2 Challenge # 1: Microarchitecture-Aware Execution-Time Modeling

Modern embedded processors have complex hardware components to speed up the average execution time of instructions. Thus, the state of such microarchitecture components directly influences the execution time of machine-code instructions and therefore the execution time of every task in the system, including the operating system. This influence has to be considered when determining upper bounds for both energy consumption and execution time.

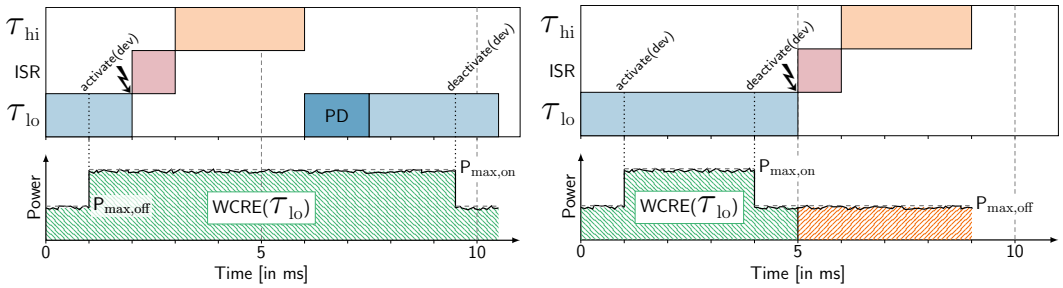


Figure 2 Preemption delays (PD) prolong execution time and also lead to higher response energy consumption. Furthermore, depending on the system's context-sensitive set of active devices, preemptions can lead to a higher response energy consumption.

Our approach: We employ well-known techniques from the literature to model the state of the pipeline and the instruction cache of our target platform at all relevant program points. Specifically, we perform cache analyses to obtain information about persistent data [8] and represent the microarchitecture state using a *microarchitecture execution graph* (MEG) [35].

2.3 Challenge # 2: Inter-Task Effects on the Microarchitecture State

Considering not only tasks in isolation but whole systems introduces another influence on the microarchitecture state: *inter-task effects*. As tasks interact with other tasks or the operating system, these interactions leave traces in the microarchitecture state. Consequently, the preempting entities disturb the microarchitecture state of the analyzed task, thereby influencing its execution time. These effects are known as *pipeline- and cache-related preemption delays* (PRPDs, CRPDs) [5, 31]. Figure 2 demonstrates how such delays contribute both to the worst-case response time (WCRT) and also the WCRE of a task in the presence of devices. Here, the low-priority task (τ_{lo}) experiences a preemption delay (PD) after being preempted by an ISR, which again leads to a resumption of the high-priority task τ_{hi}.

Our approach: To bound the number of possible preemptions, we use an existing analysis method that relies on an enumeration of possible (operating) system states [9]. We handle the costs of these preemptions with established techniques for PRPD and CRPD [5, 31] under consideration of recent findings on the effectiveness of these approaches [2, 33].

2.4 Challenge # 3: Device-Aware Energy-Consumption Modeling

To our knowledge, documentation on the microstructural energy-consumption behavior is available for neither the processor nor peripherals for any existing platform. To tackle this problem, several approaches to microstructural energy models exist to account for the number of transistor switches [21, 26, 29, 36, 38]. However, these methods fail when analyzing complex architectures and are not directly applicable to commercial off-the-shelf devices. The problem of energy-consumption modeling becomes even worse with the context-sensitive state of devices since the entire state of the system needs to be considered for the WCRE analysis: Reconsidering Figure 2, the energy consumption of τ_{lo} depends on whether the ISR occurs prior to or after activating a device after 1 ms. Consequently, a sound WCRE analysis has to be aware of possible device activations and system-wide power states.

Our approach: We overcome the challenge of missing microstructural energy models by leveraging the fact that the variances at the instruction level are minor compared to the power consumption of devices. Therefore, our approach bounds the WCEC via an *indirection* over

the execution time by performing microarchitecture-aware WCET analysis of regions that have a fixed set of active devices and thus a constant maximum power consumption P_{max}. These regions are context-sensitively determined by a system-wide path analysis and again used in a sound WCRE problem formulation [40].

3 Approach

In the following, we present our whole-system approach to tighten the WCRE estimates of tasks. We tackle the challenges introduced in Section 2 by microarchitecture-aware timing analysis of system regions with constant maximum power consumption.

3.1 Design Overview

Our approach tightens WCRE bounds by incorporating knowledge of the microarchitecture into the analysis. To achieve this, we increase the accuracy of the timing analysis by modeling an abstract microarchitecture state using microarchitecture execution graphs (MEGs) [35]. A MEG is a directed graph with nodes representing the abstract processor state (i.e., the state of the cache and the pipeline). Edges connect each node to its successor states and are weighted by the number of processor cycles it takes to complete the transition. We construct the MEG for a given instruction stream by computing the influence of each processor cycle on the microarchitecture state, thus creating nodes for each possible state.

For the pipeline, the necessary knowledge for the abstract state comprises the instructions currently processed in the pipeline and the number of CPU cycles needed to finish the processing. For the instruction cache, the interesting properties are the known cache contents and the order in which these contents are replaced in the case of a cache conflict. This analysis implies knowledge of both which datum possibly resides in which cache set as well as the replacement policy for cache ways. Having derived the necessary knowledge about pipeline and cache behavior by obtaining a hardware model and utilizing common cache analysis methods, we can construct a MEG for each basic block. The MEG enables to obtain tight bounds for the timing behavior and thus address Challenge # 1.

Constructing the MEG allows us to analyze tasks in isolation, but we are still missing the influence of inter-task effects on the microarchitecture state, as described in Challenge # 2. To consider these effects correctly, we utilize common estimation methods established in the literature [2, 31] to obtain bounds for the preemption delays in both pipeline and caches.

To address Challenge # 3, we extend the SysWCEC approach [40]. SysWCEC provides means to perform device-aware WCRE analysis of a whole system by formulating an integer linear program (ILP) over the possible (device-aware) system states. The optimization objective of the ILP is the WCRE. However, by design, the ILP also yields the execution time of the path corresponding to the WCRE of the analyzed task, which is not necessarily the same as the task's WCRT path. For example, short computations can have small or high energy consumption depending on the set of active devices. For the ILP formulation, basic blocks of the control-flow graph are aggregated to *power atomic basic blocks (PABBs)*, which represent single-entry single-exit regions that are atomic both from a scheduling perspective and regarding maximum power consumption. During the execution of a PABB no system calls are performed. Interrupts, however, can occur and potentially dispatch higher-priority tasks. Subsequently, all possible transitions between PABBs are context-sensitively enumerated and thereby inserted into the *power-state–transition graph (PSTG)*. The explicit enumeration also contains transitions for possible interrupts. Explicit path enumerations in the context of WCET analysis are often problematic due to the state-explosion problem [22]. However, for

system states, this explicit enumeration for a real-world benchmark (i.e., UAV platform), containing around 10 000 states, is conducted within a few seconds [9]. As the PSTG keeps track of the state of all devices and their power consumption, it represents a global, context-sensitive, power-aware control-flow graph, which enables to perform device-aware WCRE analysis. For further details on the PSTG and SysWCEC, we refer to previous work [40].

With all possible transitions, including interrupts, encoded in the PSTG, we bound the influence of an interrupt on a specific PSTG node. Each PSTG node again references the MEGs for each basic block it comprises. After determining the PRPD and CRPD induced by each possible interrupt for each PSTG node, we incorporate these preemption delays as additional constraints in the ILP of the SysWCEC approach. With this extension, the preemption delays of all interrupts are considered. By modeling the microarchitecture state, we are able to perform device-aware WCRE analyses with microarchitecture awareness through an indirection over a microarchitecture-aware WCET analysis. To summarize, the cost of each PSTG node i is expressed as $WCEC_i = WCET_{\mu,i} \cdot P_{max,i}$, where $WCET_{\mu,i}$ denotes the microarchitecture- and PD-aware WCET and $P_{max,i}$ the node's context-sensitive maximum power consumption. Note that this equation only holds for single PSTG nodes, which have a constant maximum power consumption. For the final WCRE determination, we formulate an ILP that contains these context-sensitive WCEC costs of PSTG nodes.

3.2 Implementation

The target platform for the analysis prototype is the Infineon XMC4500 development board [19], which features an ARM Cortex-M4 processor, as described in Section 2.1. We assume interrupt release jitter to be zero, to rule out the possibility that the analyzed scenario is interrupted by an ISR that occurred before the start of the scenario. Currently, we do not model inter–basic-block effects, which is part of our future work. Therefore, the analysis of every basic block starts with an empty pipeline. The cache states before and after the execution of a basic block are determined using `may-based persistence` analysis [8].

To bound the influence of interrupts on a PSTG node, we identify the point with the maximum PRPD among all instructions of the basic blocks associated with that PSTG node. For the CRPD, we compute the set union of *evicting cache blocks* (ECBs) for each interrupt and all tasks these interrupts potentially dispatch. We calculate the CRPD using an ECB-only approach, as recent evaluations have shown that it yields satisfactory results [33]. Both PRPD and CRPD are not integrated into the MEG, but rather considered as additional delays in the ILP and related to their corresponding interrupt occurrence.

Implementation Limitations: To bound the energy consumption associated with preemption delay, we currently assume the highest possible power state the preempted PABB could ever be executed in as the power state for the preemption delay. As it is possible to extract all possible successor states by searching the PSTG and finding the maximum possible power state, at which execution could continue, we consider this aspect as future work. Furthermore, the implementation does not include the semantics of ARM's assembly instruction for switching task contexts. As this code comprises only a few processor cycles, the influence of this code is minor compared to the overall energy consumption. Considering this aspect as well as bounding the influence of barrier instructions is part of our future work.

Our implementation is an extension to the SysWCEC [40] approach using the LLVM-based [23] analysis toolkit Platin [18, 30], which was originally developed for the PATMOS architecture [32]. Provided with a model of the target microarchitecture and the extended PSTG, Platin is now capable of generating an ILP including both microarchitecture-aware timing costs and constraints, which eventually reduce pessimism of the WCRE bounds.

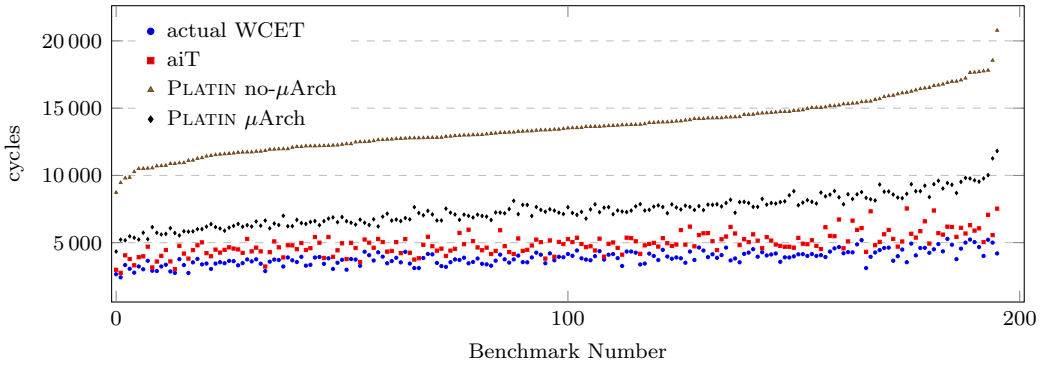


Figure 3 Actual WCET value and the WCET estimates of aiT and Platin with and without microarchitecture awareness for 196 generated benchmarks.

4 Evaluation

In this section, we present and discuss the experimental results obtained using our prototype. In Section 4.1, we first evaluate the influence of the microarchitecture awareness on the results of WCET analysis. We then set in relation the energy-consumption bounds estimated by our approach to both the actual WCRE as well as a bound computed from an approach that misses knowledge about the microarchitecture. Consequently, to obtain sound worst-case bounds, this approach has to make several pessimistic assumptions, for example, for the preemption delay. We describe the general evaluation setup for the WCRE analysis in Section 4.2 and evaluate power-consumption characteristics of our target system (see Section 4.3). We provide the results of WCRE analysis in Section 4.4 and additionally evaluate the applicability of our approach in Section 4.5.

4.1 WCET Analysis of Generated Benchmarks

To assess the accuracy gained by incorporating microarchitectural influences into timing analysis, the computed WCET of two versions of Platin, one with microarchitecture awareness and one without, are compared. For this, we generate benchmarks using the GenE benchmark generator [41], which provides automatically generated benchmarks with known WCET paths. For comparing the measurements and the static analyses (Platin no-μArch, Platin μArch), we use GenE's surrounding evaluation framework Aladdin [11]. Since Aladdin has integration for the commercially available WCET analysis tool aiT [1] on the ARM Cortex-M4, we also use these reported upper bounds for a comprehensive comparison.

Figure 3 shows the actual WCET value of the generated benchmarks obtained by measuring the known worst-case path in comparison to the analysis results. The microarchitecture-aware approach yields results, which are 39 % to 51 % smaller than the results of the approach without microarchitecture awareness. We are thus able to improve the accuracy of the analysis results by considering microarchitectural influences during the analysis. The additional comparison with aiT, an analysis tool with accurate hardware modeling as a defining feature, shows that there is still room for improvement, as in the worst-case Platin yields bounds 106 % larger than the bounds by aiT. Nonetheless, there are also benchmarks for which the Platin results are on par with the results obtained by aiT, being only 11 % larger.

4.2 Evaluation Setup for WCRE Analysis

As described in Section 3.2, the target platform for our prototype and thus the following evaluations is the Infineon XMC4500 development board [19]. We emulate the existence of devices by using four 56 Ω resistors as load. These resistors are driven by 3.3 V via external transistors, which the platform controls using GPIO pins. We measure the voltage drop over a 0.51 Ω shunt resistor and additionally over the whole system using a Tektronix MSO4034 oscilloscope with a sampling rate of 2.5 GS/s.

For the experiments, we use generated, OSEK-compliant task sets, which toggle I/O pins connected to the resistors. This way, we emulate realistic task sets interacting with devices. As a generator, we now employ GenEE [12] that creates benchmarks with a known worst-case path for energy consumption, which leads to the benchmark's WCEC. The benchmark-generation approach allows us to measure the energy consumption of that worst-case path and thus to compare the analysis results against this known baseline (i.e., actual WCEC).

4.3 Power Trace of Target System

We first examine the power-consumption behavior of our target platform. Figure 4 shows a comparison of the power consumption of one generated task set with and without attached electrical load. For the baseline trace without load (depicted below in black), we have an average power consumption of 499 mW with a standard deviation of 5 mW. These fluctuations are small in comparison to the changes in power consumption when switching devices on and off (depicted above in green). This observation underlines our approach to model the energy consumption via an indirection over microarchitecture-aware WCET analysis ($WCEC = P_{max} \cdot WCET$), where P_{max} denotes the maximum power consumption in states with a fixed set of active devices. To evaluate the overestimation of this abstraction from the dynamic power behavior, we observed the CPU's minimum and maximum power. Here, we found power variances from $P_{min} = 481$ mW up to $P_{max} = 524$ mW, which means a maximum dynamic power variation of 43 mW. Thus, always assuming a maximum constant power of the CPU leads to a worst-case overestimation of 8 %. We consider this worst-case overestimation as minor compared to the impact of transceivers in the range of hundreds of mW. Furthermore, for a reliable determination of P_{max} the ambient temperature needs to be also considered, which goes beyond the scope of this paper. Since we measure the entire system including all peripherals, parasitic capacitance, such as from the driving transistors, reduce the power consumption's transition steepness. That is, when switching on a device, it takes several microseconds until the maximum power of the new power state is reached.

4.4 WCRE Analysis With Devices

To assess the validity of our approach, we perform WCRE analysis on generated task sets with a known worst-case path for energy consumption. This way, we can compare the WCRE bound obtained with our approach both against a known baseline as well as against approaches that have to make pessimistic assumptions. This comparison allows us to evaluate the quality of our WCRE estimate compared to the actual WCRE value. Additionally, we assess the improved tightness of the analysis result gained by knowledge of the microarchitecture.

We generated task sets comprising between three and nine tasks with different execution times. Figure 5a shows the actual WCRE values of these task sets as well as the analysis results of our approach and two pessimistic approaches. One of these approaches is SysWCEC without microarchitecture awareness (labeled *pessimistic SysWCEC*), which is forced to make

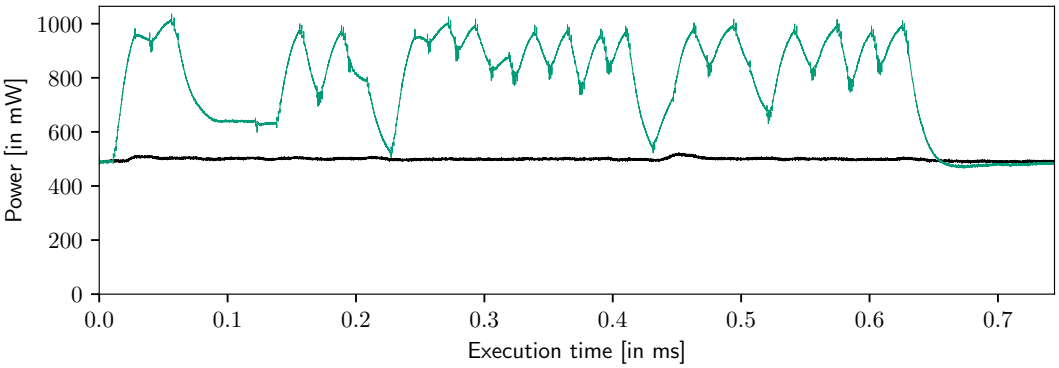


Figure 4 Trace of the power consumption of one evaluation benchmark with (green, above) and without (black, below) attached devices. The toggling of devices entails power-consumption changes which are much larger than fluctuations observed due to microarchitecture effects.

pessimistic assumptions, for example, for instruction fetch latencies (i.e., cache misses). The other (labeled *µArch always-on*) is a microarchitecture-aware approach lacking device-state modeling, which thus has to assume that all devices are always on.

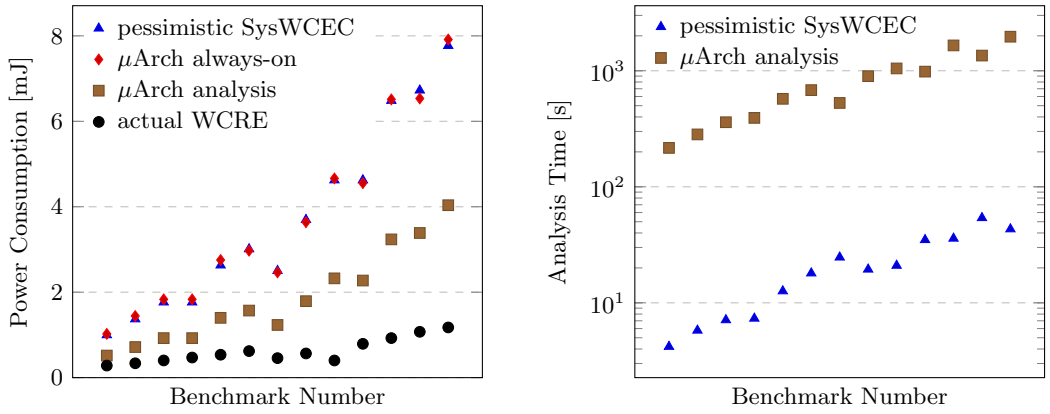
Our approach yields overestimations of 83 % up to 479 %, with a median of 170 % and a geometric mean of 173 % (an overestimation of 0 % represents the actual WCRE). While bringing potential for improvements to light, the results show that by leveraging both microarchitecture awareness and knowledge about the device states, we are able to give tighter bounds in comparison to the pessimistic approaches. We achieve between 46 % and up to 51 % tighter bounds compared to the analysis without the microarchitecture model, and between 49 % and up to 52 % tighter bounds compared to the always-on approach. With overestimations of around 170 %, our approach is capable of providing acceptable bounds on the WCRE, while yielding significantly better results than both pessimistic approaches. This confirms the validity of our approach and demonstrates the necessity to consider both the microarchitecture state and the state of devices to achieve tight WCRE bounds. The results further support our solution to circumvent the microarchitecture modeling of energetic behavior through an indirection over execution time analysis.

4.5 Applicability and Scalability

In the previous section, we showed that our approach is capable of computing tighter results. This improvement comes, however, at the cost of increased analysis times due to the complexity of modeling the microarchitecture state. Figure 5b depicts the execution times of the WCRE analyses on an Intel i5-4590 CPU with 16 GB of RAM. PLATIN without microarchitecture awareness finishes the analysis in a few seconds, whereas the microarchitecture-aware approach takes up to 50 x longer. However, the analysis times are still in the range of minutes for benchmarks with up to 20 000 processor cycles. We argue that this increase is acceptable for the up to 51 % tighter results gained by microarchitecture awareness.

5 Related & Future Work

To the best of our knowledge, this approach is the first for WCRE in cyber-physical systems where microarchitecture-aware timing analysis is exploited to account for on/off timespans of devices. There exists a large body of related work on microarchitecture-aware energy modeling [21, 26, 29, 36, 38]. In contrast to these approaches, we do not directly consider



(a) Actual WCRE value (circles) and the estimates of PLATIN with (squares) and without (triangles) microarchitecture awareness, and an always-on approach (diamonds).

(b) Analysis time needed with (squares) and without (triangles) microarchitecture awareness to compute the WCRE estimates for the generated benchmarks shown on the left.

Figure 5 Results of the WCRE analysis for 13 generated benchmarks.

dynamic power variations due to the number of transistor switches on the microarchitectural level. In this context, Morse et al. presented an overview of the limitations of WCEC modeling on the microarchitecture level [26]. The overview outlines the problem of data-dependent power consumption of instructions. For processors with limited microarchitectural complexity, we found power-consumption variances being only a minor fraction of the overall power consumption of device-driven systems [40], where active transceivers consume hundreds of mW. Nevertheless, approaches to reduce the pessimism of the processor, as proposed by Morse et al. [26], are directly applicable to our approach. Furthermore, we see potential to model data-depending operations again with existing WCET approaches for data-flow analysis [4]. This data-dependent analysis potentially improves modeling transistor switches.

A further aspect of future work is handling devices that, for example, reduce their energy consumption after a defined timespan once activated. Since our approach is aware of (microarchitecture-aware) WCETs along system-wide paths, we are able to apply functions for the energy-consumption cost to account for these complex energy-consumption behaviors.

Additionally, we believe that we can further reduce the pessimism of the WCRE analysis results by including inter–basic-block and inter-PABB effects in the analysis, for example, by modeling pipeline interleaving and performing data-flow analyses on the PSTG.

6 Conclusion

Static worst-case energy-consumption analysis gains importance with the increasing number of safety-critical cyber-physical systems that are battery-operated. To guarantee safe operation, determining reliable and precise WCRE bounds that account for the whole system is crucial.

This paper presents an approach that decomposes the system into blocks with a fixed set of active devices, which is used to determine system-wide control flows. A microarchitecture-aware WCET analysis along these paths precisely determines the durations devices are active under consideration of preemption delays. This information is used as costs in a WCRE formulation. Our evaluations show that this composite analysis significantly reduces analysis pessimism. The source code of our analysis approach is publicly available.

Source code: `https://gitlab.cs.fau.de/syswcec-uarch`

References

1 AbsInt. aiT WCET analyzers. `https://www.absint.com/ait/`.

2 S. Altmeyer, R. I. Davis, and C. Maiza. Improved cache related pre-emption delay aware response time analysis for fixed priority pre-emptive systems. *Real-Time Systems*, 48(5):499–526, 2012.

3 C. Ballabriga, H. Cassé, C. Rochange, and P. Sainrat. OTAWA: An open toolbox for adaptive WCET analysis. In *Proc. of the 8th Int'l Work. on Software Technolgies for Embedded and Ubiquitous Systems (SEUS '10)*, pages 35–46, 2010.

4 Johann Blieberger. Data-flow frameworks for worst-case execution time analysis. *Real-Time Systems*, 22(3):183–227, 2002.

5 J. V. Busquets-Mataix, J. J. Serrano, R. Ors, P. Gil, and A. Wellings. Adding instruction cache effect to schedulability analysis of preemptive real-time systems. In *Proc. of the 2nd Real-Time Technology and Applications Symp. (RTAS '96)*, pages 204–212, 1996.

6 F. Cassez, R. Rydhof Hansen, and M. C. Olesen. What is a timing anomaly? In *Proc. of the 12th Int'l Work. on Worst-Case Execution Time Analysis (WCET '12)*, 2012.

7 Shenzhen HOPE Microelectronics Co. RFM95/96/97/98(W) - low power long range transceiver module, 2019.

8 C. Cullmann. Cache Persistence Analysis: Theory and Practice. *ACM Trans. on Embedded Computing Systems (ACM TECS)*, 12(1s):40:1–40:25, 2013.

9 C. Dietrich, P. Wägemann, P. Ulbrich, and D. Lohmann. SysWCET: Whole-system response-time analysis for fixed-priority real-time systems. In *Proc. of the 23nd Real-Time and Embedded Technology and Applications Symp. (RTAS '17)*, pages 37–48, 2017.

10 N. Duda, T. Nowak, M. Hartmann, M. Schadhauser, B. Cassens, P. Wägemann, M. Nabeel, S. Ripperger, S. Herbst, K. Meyer-Wegener, F. Mayer, F. Dressler, W. Schröder-Preikschat, R. Kapitza, J. Robert, J. Thielecke, R. Weigel, and A. Koelpin. BATS: adaptive ultra low power sensor network for animal tracking. *Sensors*, 18(10):1–34, 2018.

11 C. Eichler, P. Wägemann, T. Distler, and W. Schröder-Preikschat. Demo Abstract: Tooling Support for Benchmarking Timing Analysis. In *Proc. of the 23rd Real-Time and Embedded Technology and Applications Symp. (RTAS '17)*, pages 159–160, 2017.

12 C. Eichler, P. Wägemann, and W. Schröder-Preikschat. GenEE: A benchmark generator for static analysis tools of energy-constrained cyber-physical systems. In *Proc. of the 2nd Work. on Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench '19)*, 2019. URL: `https://www4.cs.fau.de/Publications/2019/eichler_19_cpsiotbench.pdf`.

13 Freescale Semiconductor, Inc. *KL46 Sub-Family Reference Manual*, 2013.

14 Freescale Semiconductor, Inc. *Kinetis KL46 Sub-Family*, 2014.

15 Fujitsu Semiconductor. *FRAM MB85RC256V*, 2013.

16 G. Gebhard. Timing anomalies reloaded. In *Proc. of the 10th Int'l Work. on Worst-Case Execution Time Analysis (WCET '10)*, 2010.

17 D. Hardy, B. Rouxel, and I. Puaut. The Heptane Static Worst-Case Execution Time Estimation Tool. In *Proc. of the 17th Int'l Work. on Worst-Case Execution Time Analysis (WCET '17)*, pages 8:1–8:12, 2017.

18 S. Hepp, B. Huber, D. Prokesch, and P. Puschner. The platin Tool Kit – The T-CREST Approach for Compiler and WCET Integration. In *Proc. of the 18th Kolloquium Programmiersprachen und Grundlagen der Programmierung (KPS '15)*, pages 277–292, 2015.

19 Infineon Technologies AG. Evaluation Board XMC4500 Relax Kit & XMC4500 Relax Lite Kit, 2014.

20 Infineon Technologies AG. *XMC4500 Microcontroller Series for Industrial Applications — Data Sheet*, 2017. V1.5 2017-12.

21 R. Jayaseelan, T. Mitra, and X. Li. Estimating the Worst-Case Energy Consumption of Embedded Software. In *Proc. of the 12th Real-Time and Embedded Technology and Applications Symp. (RTAS '06)*, pages 81–90, 2006.

22 J. Knoop, L. Kovács, and J. Zwirchmayr. WCET squeezing: On-demand feasibility refinement for proven precise WCET-bounds. In *Proc. of the 21st Int'l Conf. on Real-Time Networks and Systems (RTNS '13)*, pages 161–170, 2013.

23 C. Lattner and V. Adve. LLVM: A compilation framework for lifelong program analysis & transformation. In *Proc. of the Int'l Symp. on Code Generation and Optimization (CGO '04)*, pages 75–86, 2004.

24 Xianfeng Li, Yun Liang, Tulika Mitra, and Abhik Roychoudhury. Chronos: A timing analyzer for embedded software. *Science of Computer Programming*, 69(1):56–67, 2007.

25 B. Lisper. SWEET – A tool for WCET flow analysis. In *Proc. of the 6th Int'l Symp. On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA '14)*, pages 482–485, 2014.

26 J. Morse, S. Kerrison, and K. Eder. On the Limitations of Analyzing Worst-Case Dynamic Energy of Processing. *ACM Trans. on Embedded Computing Systems (ACM TECS)*, 17(3):59:1–59:22, 2018.

27 OSEK/VDX Group. OIL: OSEK implementation language. Technical report, OSEK/VDX Group, July 2004.

28 OSEK/VDX Group. Operating System Specification 2.2.3. Technical report, OSEK/VDX Group, February 2005.

29 J. Pallister, S. Kerrison, J. Morse, and K. Eder. Data Dependent Energy Modeling for Worst Case Energy Consumption Analysis. In *Proc. of the 20th Int'l Work. on Software and Compilers for Embedded Systems (SCOPES '17)*, pages 51–59, 2017.

30 P. Puschner, D. Prokesch, B. Huber, J. Knoop, S. Hepp, and G. Gebhard. The T-CREST Approach of Compiler and WCET-Analysis Integration. In *Proc. of the 9th Work. on Software Technologies for Future Embedded and Ubiquitious Systems (SEUS '13)*, pages 33–40, 2013.

31 J. Schneider. Cache and Pipeline Sensitive Fixed Priority Scheduling for Preemptive Real-Time Systems. In *Proc. of the 21st Real-Time Systems Symp. (RTSS '00)*, pages 195–204, 2000.

32 M. Schoeberl et al. T-CREST: Time-predictable multi-core architecture for embedded systems. *Journal of Systems Architecture*, 61:449–471, 2015.

33 D. Shah, S. Hahn, and J. Reineke. Experimental Evaluation of Cache-Related Preemption Delay Aware Timing Analysis. In *18th Int'l Work. on Worst-Case Execution Time Analysis (WCET '18)*, pages 7:1–7:11, 2018.

34 Silicon Laboratories, Inc. *EFM32GG Reference Manual*, 2016.

35 I. J. Stein. *ILP-based path analysis on abstract pipeline state graphs.* epubli, 2010.

36 S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel. An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations. In *Proc. of the Int'l Work. on Power And Timing Modeling, Optimization and Simulation (PATMOS '01)*, 2001.

37 Espressif Systems. ESP8266EX Datasheet Version 6.0, 2018.

38 V. Tiwari, S. Malik, A. Wolfe, and M. T.-C. Lee. Instruction level power analysis and optimization of software. *Journal of VLSI Signal Processing Systems*, 13(2-3):223–238, 1996.

39 M. Völp, M. Hähnel, and A. Lackorzynski. Has Energy Surpassed Timeliness? Scheduling Energy-Constrained Mixed-Criticality Systems. In *Proc. of the 20th Real-Time and Embedded Technology and Applications Symp. (RTAS '14)*, pages 275–284, 2014.

40 P. Wägemann, C. Dietrich, T. Distler, P. Ulbrich, and W. Schröder-Preikschat. Whole-System Worst-Case Energy-Consumption Analysis for Energy-Constrained Real-Time Systems. In *Proc. of the 30th Euromicro Conf. on Real-Time Systems (ECRTS '18)*, pages 24:1–24:25, 2018. doi:10.4230/LIPIcs.ECRTS.2018.24.

41 P. Wägemann, T. Distler, C. Eichler, and W. Schröder-Preikschat. Benchmark Generation for Timing Analysis. In *Proc. of the 23rd Real-Time and Embedded Technology and Applications Symp. (RTAS '17)*, pages 319–330, 2017.

42 P. Wägemann, T. Distler, H. Janker, P. Raffeck, V. Sieh, and W. Schröder-Preikschat. Operating Energy-Neutral Real-Time Systems. *ACM Trans. on Embedded Computing Systems (ACM TECS)*, 17(1):11:1–11:25, 2017.