Embedded Systems Under Resource Limits

Eingebettete Systeme unter Ressourcenbeschränkungen

Dr.-Ing. Peter Wägemann

Der Technischen Fakultät der
Friedrich-Alexander-Universität Erlangen-Nürnberg

als

HABILITATIONSSCHRIFT

Erlangen — 2026

Als Habilitation genehmigt von der
Technischen Fakultät der
Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der Einreichung: 31. Dezember 2025
Erteilung der Lehrbefähigung: 20. Mai 2026

Fachmentorat: Prof. Dr.-Ing. Rüdiger Kapitza
Friedrich-Alexander-Universität Erlangen-Nürnberg
(ab 1. April 2024)

Prof. Dr.-Ing. habil. Wolfgang Schröder-Preikschat
Friedrich-Alexander-Universität Erlangen-Nürnberg
(bis 31. März 2024, Ruhestand)

Prof. Dr.-Ing. Felix Freiling
Friedrich-Alexander-Universität Erlangen-Nürnberg

Prof. Dr. Michael Engel
Otto-Friedrich-Universität Bamberg

Gutachter: Prof. Dr.-Ing. Jan Reineke
Saarland University

Prof. Dr.-Ing. Marcus Völp
University of Luxembourg

ABSTRACT

The vast majority of hardware platforms are integrated into embedded systems, making them a strong driver for innovations, for example, in the health, energy, and mobility sectors. Many types of embedded systems are characterized by their stringent resource constraints: While the software of embedded real-time systems must fulfill timing constraints and guarantee that deadlines are met, battery-operated or batteryless/energy-harvesting devices similarly demand reliable execution under strict energy budgets. Likewise, limited memory creates challenges for running applications with substantial code and data requirements. With a generic view on the topic of resource limits in embedded systems, this cumulative habilitation treatise further aims to advance more sustainable, carbon-aware systems. Within these four types of resource constraints (i.e., time, energy, memory, and carbon), a main focus of this treatise is on providing provably safe execution for target applications while minimizing the resources required for this execution.

Targeting these four types of resource constraints, I have concentrated on three overlapping subject areas over the last years with the eventual goal of making embedded systems more reliable and/or more efficient. These areas comprise (1) analysis and (2) optimization techniques for (3) the design and implementation of embedded systems that execute within resource limits.

This cumulative habilitation treatise, which covers 12 peer-reviewed papers, first describes static and dynamic program analysis techniques to determine worst-case behaviors of program code. Second, based on previously identified analysis results, optimization techniques exploit tradeoffs in order to find resource-optimized configurations of systems. Third, this work presents several embedded systems where both the analysis and optimization techniques are utilized. The development of techniques and abstractions that comprehensively address the three overlapping subject areas is the central goal of this treatise.

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Domain of Embedded Systems

Constraints of Embedded Systems Studies report that around $90\,\%$ to $98\,\%$ [64, 65] of hardware platforms are used for embedded systems. These types of systems are embedded into a specific environment, fulfill a specific purpose, and are ubiquitous in a technology-driven society. The main scope of this cumulative habilitation treatise comprises embedded systems that have soft or hard constraints with regard to their (1) temporal behavior, (2) energy demand, (3) memory consumption, (4) embodied carbon footprint, or a combination of these aspects.

Ubiquity of Embedded Systems In the early 1990s, Mark Weiser described "The Computer for the 21st Century" and thereby coined the idea of *ubiquitous computing* [66]. Weiser predicted that computing will happen everywhere and anytime with embedded systems[1]. Around 35 years later, numerous features along with connectivity support are nowadays integrated into single systems, for example, in view of the availability of smartphones. Besides these mobile devices, which can increasingly serve general computing scenarios, a report of the Internet of Things (IoT), as of the end of 2025, estimates approximately 21 billion connected devices [67]. The Internet of Batteryless Things (IoBT) [68] or the Internet of Medical Things (IoMT) can lead to novel types of embedded systems in society.

Safe & Efficient Operation While embedded systems for consumer electronics often fulfill rather uncritical services, numerous safety-critical application scenarios exist. Medical devices, such as medical implants, serve as an example of embedded systems with both

[1] Interestingly, Weiser mentioned that "No revolution in artificial intelligence is needed, [...]". Given that, around 35 years later, numerous embedded systems are driven by machine-learning techniques, the currently ongoing trend of artificial intelligence likely exceeds Weiser's vision due to, for example, the possibility of having a voice-controlled interaction with embedded systems.

timing and energy-related restrictions: Due to the limited available energy resources, these systems should have a small energy demand while likewise guaranteeing timeliness, when, for example, monitoring health states. Generalizing from this example, one objective of several subsequent works is to (a) reliably (i.e., within resource limits) operate systems while (b) reducing the required resources to achieve this operation. In short, both analysis and optimization techniques represent the foundation for reliable and resource-optimized system operation.

Journey with Embedded Systems Since my initial steps in academia [57] with energy-constrained embedded systems, I have been researching embedded systems for more than ten years. Driven by my still lasting fascination for these "invisible" computing systems, this core research domain still remained, and I feel very honored to step-wise advance the state of the art in this domain together with my research group "Embedded Systems Software" and the research community in general. In this domain of embedded systems, this presented cumulative habilitation treatise subsequently introduces and outlines several works, which were all formally published in the years 2023, 2024, and 2025.

1.2 Resource Limits & Tradeoffs in Embedded Systems

Embedded systems usually come with inherent resource limits, since they should optimally serve a specific use case and are, consequently, tailored towards this use case. This is in contrast to non-tailored and general-purpose systems, serving generic use cases. Some resource limits are inherent to specific systems: For example, in this treatise, *TinyBFT* [10] describes a scenario where the entire software stack along with the application and a protocol for Byzantine fault tolerance (BFT) are required to stay within the available memory limits (i.e., $400\,\mathrm{kB}$ of RAM) for reliable operation. In this setting, no tradeoff exists with regard to the hard memory limit. With a different objective of handling limits and tradeoffs, the work on *Crêpe* [7] addresses the resources of time, energy, and memory. The goal of *Crêpe* is to find solutions that reduce the energy demand while accounting for given deadlines (i.e., hard timing limits). Further, *Crêpe*'s way of handling scheduling decisions exploits a tradeoff between increased memory demand for reduced time/energy penalties, which, like *TinyBFT*, has to stay within the system's available memory. In general, this cumulative habilitation treatise covers the four mentioned resource constraints along with several of their tradeoffs. Thereby, the energy constraints describe the analysis and management of *operational* energy resources, while the carbon constraints concern the *embodied* carbon resources of embedded systems. Figure 1.1 outlines these four resources along with their considered overlaps in this treatise's summarized publications.

This manuscript is based on 12 peer-reviewed publications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Thus, the first 12 numbered references indicate the central papers of this cumulative treatise. The reprints of these papers are included in Appendix B, and the detailed treatise-related publications list is shown in Appendix A.2. My personal contributions to these 12 papers are outlined in Section 1.3. These works are subdivided into the overarching and interconnected three areas of resource-demand analysis (see Chapter 3),

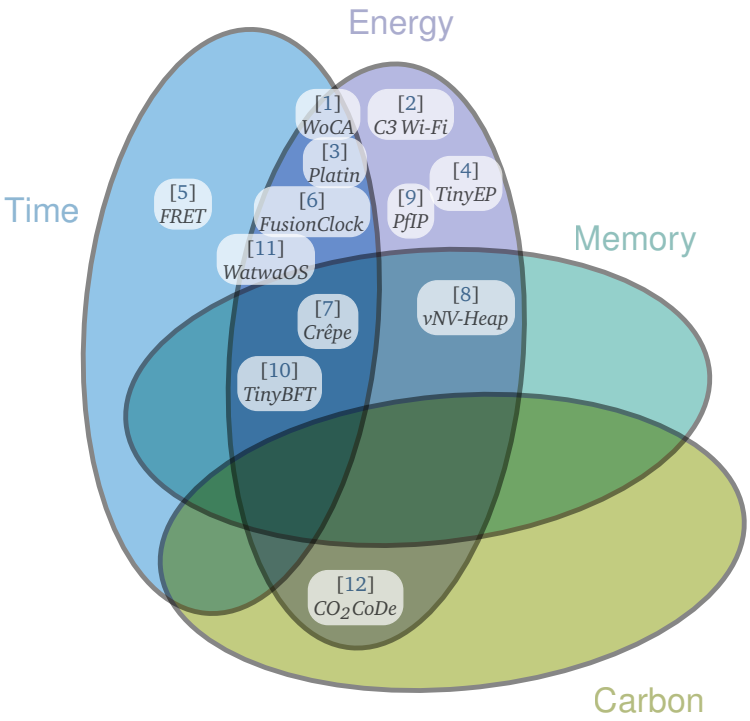

Figure 1.1: The summarized publications of this treatise address analyses, optimizations, and systems themselves within resource limits. The four dimensions of resource constraints are time, energy, memory, and (embodied) carbon. Exploiting tradeoffs requires comprehensive considerations of two or more overlapping resource types.

resource-optimization techniques (see Chapter 4), and the design and implementation of the systems themselves (see Chapter 5). The core structure of this document, along with the respective publications, is listed as follows:

Chapter 3: Static & Dynamic Program Analysis

LCTES '24 Phillip Raffeck, Johannes Maier, and Peter Wägemann [1]
WoCA: Avoiding Intermittent Execution in Embedded Systems by Worst-Case Analyses with Device States
Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '24)

ENSsys '25 Ishwar Mudraje, Jasper Devreker, Kai Vogelgesang, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, and Thorsten Herfet [2]
Reverse Engineering the ESP32-C3 Wi-Fi Drivers for Static Worst-Case Analysis of Intermittently-Powered Systems
Proceedings of the 13th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems (ENSsys '25)

WCET '24 Emad Jacob Maroun, Eva Dengler, Christian Dietrich, Stefan Hepp, Henriette Herzog, Benedikt Huber, Jens Knoop, Daniel Wiltsche-Prokesch, Peter Puschner, Phillip Raffeck, Martin Schoeberl, Simon Schuster, and Peter Wägemann [3]
The Platin Multi-Target Worst-Case Analysis Tool
Proceedings of the 22nd International Workshop on Worst-Case Execution Time Analysis (WCET '24)

Access '24 Kilian Müller, Johannes Weidner, Norman Franchi, and Peter Wägemann [4]
TinyEP: TinyML-enhanced Energy Profiling for Extreme Edge Devices
IEEE Access

RTSS '25 Alwin Berger, Simon Schuster, Peter Wägemann, and Peter Ulbrich [5]
Dynamic Fuzzing-Based Whole-System Timing Analysis
Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)

Chapter 4: Optimizing Resource Demands

ECRTS '23 Eva Dengler, Phillip Raffeck, Simon Schuster, and Peter Wägemann [6]
FusionClock: Energy-Optimal Clock-Tree Reconfigurations for Energy-Constrained Real-Time Systems
Proceedings of the 35th Euromicro Conference on Real-Time Systems (ECRTS '23)

ECRTS '24 Eva Dengler and Peter Wägemann [7]
Crêpe: Clock-Reconfiguration–Aware Preemption Control in Real-Time Systems with Devices
Proceedings of the 36th Euromicro Conference on Real-Time Systems (ECRTS '24)

Chapter 5: Utilizing Analyses & Optimizations for Systems

LCTES '25 Markus E. Gerber, Luis Gerhorst, Ishwar Mudraje, Kai Vogelgesang, Thorsten Herfet, and Peter Wägemann [8]
vNV-Heap: An Ownership-based Virtually Non-Volatile Heap for Embedded Systems
Proceedings of the 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2025)

CCNC '25 Kai Vogelgesang, Ishwar Mudraje, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, Thorsten Herfet, and Wolfgang Schröder-Preikschat [9]
PfIP: A UDP/IP Transactional Network Stack for Power-Failure Resilience in Embedded Systems
Proceedings of the 22nd IEEE Consumer Communications & Networking Conference (CCNC 2025)

RTAS '24 Harald Böhm, Tobias Distler, and Peter Wägemann [10]
TinyBFT: Byzantine Fault-Tolerant Replication for Highly Resource-Constrained Embedded Systems
Proceedings of the 30th Real-Time and Embedded Technology and Applications Symposium (RTAS '24)

RTSS '25 Tobias Häberlein, Eva Dengler, Phillip Raffeck, and Peter Wägemann [11]
WatwaOS: A Framework for Worst-Case–Aware Tailoring and Whole-System Analysis of Energy-Constrained Real-Time Systems
Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)

HotCarbon '24 Phillip Raffeck, Sven Posner, and Peter Wägemann [12]
CO2CoDe: Towards Carbon-Aware Hardware/Software Co-Design for Intermittently-Powered Embedded Systems
Proceedings of the 3rd HotCarbon Workshop on Sustainable Computer Systems (HotCarbon '24)

Comments on the Order of Author Names

In the above-listed papers, the first author conducted the majority of the implementation and experimental work, who was either a doctoral or master's student. This work happened in close collaboration with me, as either the only scientific leader/mentor or one of the scientific leaders/mentors. Throughout these papers, the author names are chosen in a way that the last author(s) represent the only or one of the main supervising/mentoring author(s). In collaborative projects across different universities, specifically in these publications [2, 5, 8], the last author refers to the doctoral student's main supervisor. The work on *Platin* [3] poses an exception to the order mentioned above, because the respective paper has an alphabetical sorting of author names. Furthermore, the author order of *PfIP* [9] corresponds to the *sequence-determines-credit approach (SDC)*. For the publications on *WoCA* [1], *FusionClock* [6], *Crêpe* [7], *WatwaOS* [11], and *CO_2CoDe* [12], I have served as the sole supervising author.

1.3 Outline of Papers & Own Contributions

While the main three chapters give further insights into the respective papers, the following sections outline the papers and highlight my personal contributions to these works.

1.3.1 Safe & Efficient Intermittent Operation with *WoCA* [1]

In the domain of energy-constrained embedded systems, the IoBT [68] has the potential to change our current understanding of embedded systems with respect to their power supply and, thereby, their ubiquity: These embedded "things" harvest their entire energy from the environment (e.g., through solar cells). This energy-self-sufficient operation avoids the need to change batteries and comes with the vision of systems with an infinite operational phase. For such types of systems, the work of *WoCA* [1] presents an analysis approach that guarantees the uninterrupted execution of device operations with transactional semantics (e.g., sending a packet via a transceiver device). The main novelty of *WoCA* is the use of device-related graphs, which accurately describe device states and transitions, within a static worst-case analysis approach. Central ideas for using worst-case analysis techniques for the domain of intermittent computing (i.e., computing with an intermittent

power supply) originate from my previous works in this area on analyses [23, 28] and systems [27, 33]. For the work on *WoCA*, I have been the only supervisor. The first author of this paper, Phillip Raffeck, received his doctorate [63] under my supervision, in which the work on the *WoCA* approach plays a relevant role.

1.3.2 *ESP32-C3-OpenMAC*: Wi-Fi Drivers for Static Analysis [2]

Accurate static analysis inevitably requires knowledge of the hardware and software. For these considerations, this treatise uses the attribute *whole-system* to describe the entire system's stack with all its layers, starting from the hardware, through the operating system (OS) with its device drivers and scheduler, to the application layer. Unfortunately, for relevant embedded systems, the drivers for accessing devices are often not publicly available as source code. To solve this problem for the Wi-Fi stack of the RISC-V–based ESP32-C3 platform, Mudraje et al. [2] present reverse-engineered Wi-Fi drivers, based on the preliminary work of *OpenMAC* [69, 70]. For this work, which was conducted in the scope of the DFG-funded project *Resilient Power-Constrained Embedded Communication Terminals* (*ResPECT*) between Saarland Informatics Campus (SIC) and Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), I was one of the scientific project leads.

1.3.3 The Open-Source Worst-Case Analysis Framework *Platin* [3]

Conducting worst-case analysis of program code is a challenging endeavor since it requires knowledge of various layers and aspects in embedded systems, namely, (1) programming languages & compiler infrastructure, (2) computer architectures, (3) mathematical problem formulations, (4) operating systems, and (5) modeling application behavior. In addition to these relevant aspects for determining the worst-case execution time (WCET), for estimates of the worst-case energy consumption (WCEC), the further considerations of (6) system-level power management (inside the computing systems) and (7) the power behavior of electronics are necessary. To address these challenges, the *Platin* [3] framework provides an open-source framework for research in this domain. Numerous researchers have been contributing to this framework, and the respective publication [3] gives an overview of contributions from various groups. I have been one of the mentoring authors for this paper and contributed to this framework with several co-authored works [1, 20, 22, 23, 25, 26, 32, 36].

1.3.4 *TinyEP* for Online Environment-Aware Energy Assessment [4]

Depending on the application scenario, the source code is not always available for an ahead-of-runtime analysis. However, assessing the energy demand of executed jobs, whose code is unknown, on embedded systems is crucial for making energy-aware scheduling decisions or for accurately accounting in multi-tenant systems. One problem for precise accounting is the energy-related overhead of the involved energy-measurement equipment when the system is deployed in the field. This is especially challenging under varying environmental conditions (i.e., changing temperatures), which affect the systems' energy

demand. To solve these problems, *TinyEP* [4] introduces an approach based on machine learning: Specifically, *TinyEP* involves the training of a deep neural network (DNN) to learn the power-demand behavior of targeted applications. Lightweight online inferences, driven by frameworks like TinyML [71, 72], allow the assessment of energy demand (i.e., power over time) of tasks. The master's thesis of Johannes Weidner [73], for which Kilian Müller and I were responsible for the content-related supervision, contributed to the *TinyEP* approach.

1.3.5 Dynamic Response-Time Analysis with *FRET* [5]

Having accurate execution-time estimates is of high relevance for temporal budgeting in real-time systems. While static analysis can determine provable upper bounds of the actual WCET, this type of analysis is not feasible for many modern and highly complex computer architectures. As a resort, dynamic analysis techniques are a practical approach to determine estimates of the WCET, which usually underestimate the actual WCET. In this context, *FRET* [5] presents an analysis that exploits fuzzing techniques. To allow *FRET* to capture the behavior between tasks, *FRET* dynamically builds a state-transition graph (STG) [26, 74], which describes intra-task dependencies. Since this STG is crucial for several works in this treatise, Section 3.2 gives an introduction to the underlying abstractions. With this knowledge of the STG, *FRET* leverages its fuzzing loop to heuristically explore novel program paths that are relevant for the worst-case response time. *FRET* is a collaborative work between Technische Universität Dortmund and FAU, and I have been one of the two scientific leaders.

1.3.6 Clock-Aware Optimizations with *FusionClock* [6]

In most cases, optimizing the resource demand in embedded systems involves a tradeoff between two (or more) properties. The resources of time and energy are linked with the tradeoff that a faster execution speed usually comes with an increased power demand. This tradeoff is configurable in many modern embedded system-on-chip (SoC) platforms with complex, intertwined clock subsystems. One problem in the domain of energy-constrained real-time systems is finding configuration sequences of the clock that (1) guarantee timeliness of all executed tasks while (2) minimizing the required energy demand for the execution. To address this problem, the *FusionClock* [6] approach combines a clock-subsystem model, with possible states and transition penalties, and static analysis techniques of the application's code to formulate a mathematical optimization problem. The solution to this problem eventually yields optimal (re-)configuration sequences of the clock subsystem with respect to deadline constraints.

1.3.7 *Crêpe*: Resource-Optimized Preemption Control [7]

The work of *Crêpe* [7] builds upon the insights of *FusionClock* and tackles the previously excluded problem of asynchronously occurring interrupts: Interrupts are usually inherent to real-world systems, for example, to communicate with devices: For instance, transceivers

release interrupts when a received packet is available and ready to be processed. Comparable to *FusionClock*, *Crêpe* has the objective to determine energy-minimal configuration sequences while accounting for given deadlines. In contrast to *FusionClock*, the *Crêpe* approach explicitly controls preemptions of running tasks through interrupts and schedules them under consideration of the clock subsystem and sleep states. The work on *Crêpe*, authored by Eva Dengler and me, received the *Best Paper Award* at the *36th Euromicro Conference on Real-Time Systems (ECRTS '24)*. Both *FusionClock* and *Crêpe* are part of the DFG-funded project *Whole-System Optimality Analysis and Tailoring of Worst-Case–Constrained Applications* (*Watwa*), of which I am the sole project leader. For these two papers, I am likewise the sole scientific supervisor.

1.3.8 Heaps in Energy-Constrained Embedded Systems with *vNV-Heap* [8]

The previously mentioned analysis techniques have the purpose of eventually serving actual systems for supporting their efficiency and/or their guaranteed execution within resource bounds. One of these systems presented in this treatise is the *vNV-Heap* approach [8], which provides the support of heap data structures in embedded systems: *vNV-Heap* targets embedded systems that face power failures, as prevailing in the mentioned IoBT: The addressed systems have to persist their state in non-volatile memory (NVM) before an impending power failure. Later, the treatise explains in Section 3.4 the required hardware support to (1) safely suspend the system to NVM before a power failure actually occurs and to (2) safely wake up the system once sufficient energy is available. *vNV-Heap* exploits the programming-language features of borrowing and ownership for precisely tracking modified state in (volatile) memory, which can make the system's checkpointing to persistent memory more energy-efficient. This work is part of the DFG-funded project *ResPECT*, for which I supervised the subproject from FAU.

1.3.9 Transactional Network Stacks Against Power Failures with *PfIP* [9]

Like *vNV-Heap*, the work on *PfIP* [9] is part of the *ResPECT* research project, whose purpose is to provide power-failure resilience to embedded communication terminals. In this context, power failures might leave the communicating system in an inconsistent state. To solve this problem, *PfIP* proposes a UDP/IP network stack with transactional semantics. The system ensures that transactions are only started when sufficient energy is available. Consequently, the system never has to rollback to a previous checkpoint and, thereby, ensures consistency. For *PfIP*, I have been one of the mentoring supervisors.

1.3.10 *TinyBFT*: Byzantine Fault Tolerance in Embedded Systems [10]

The fault model of BFT in embedded networked systems is required when these systems share a common modifiable state and require consensus over this state in the presence of arbitrary faults. Traditional BFT protocols were developed for more compute-intensive distributed systems in comparison to rather resource-constrained embedded systems with limited memory resources. The work on *TinyBFT* [10] introduced an approach based

on the seminal PBFT protocol [75]. *TinyBFT* achieves distributed consensus within the memory resource constraints of 400 kB RAM (on a RISC-V–based ESP32-C3 platform). This work on *TinyBFT* provided the foundation for the subsequent DFG-funded research project *Byzantine Fault Tolerance in Time- and Energy-constrained Applications Under Memory Bounds* (*BFTeam*), a collaborative project between Otto-Friedrich-Universität Bamberg and FAU, of which I am one of the two scientific project leaders, as is the case for the *TinyBFT* publication.

1.3.11 Whole-System Tailoring with *WatwaOS* [11]

The previously mentioned works on *FusionClock* and *Crêpe* explored the time-energy tradeoff under consideration of complex intertwined clock subsystems of modern embedded SoC platforms. However, they disregard the fact that practical embedded systems, in many cases, make use of OSes to avoid labor-intensive tailoring of applications to single hardware platforms. OSes exist that make use of the tradeoff that modern clock subsystems with multiple input clocks offer [76, 77]. However, they are not able to make resource-optimal decisions with regard to strict time and energy constraints. To solve this problem, *WatwaOS* [11] introduced both a framework and a (generated) OS for worst-case–aware tailoring and whole-system analysis of energy-constrained real-time systems. As the name suggests, *WatwaOS* is part of the DFG-funded research project *Watwa*, of which I am the sole project leader.

1.3.12 Carbon-Aware Co-Design of Embedded Systems with *CO_2CoDe* [12]

The IoBT [68] comes with the goal of transforming the existing and evolving Internet of Things into a more sustainable societal infrastructure. With regard to these systems' runtime, the carbon footprint is zero since they harvest their required energy from the environment. However, this consideration neglects the fact that the majority of the required carbon resources of mobile systems in general are part of the *embodied carbon footprint* [78]. This treatise argues that this embodied carbon footprint requires special attention when advancing to more sustainable computer infrastructures. In this line of thought, the work on *CO_2CoDe* [12] proposes an approach to carbon-aware hardware/software co-design of embedded systems under intermittent power supply. *CO_2CoDe* chooses between different components for storing energy in systems with an intermittent power supply. I have been the only supervising mentor of the work on *CO_2CoDe*.

1.4 Open-Source Availability of Prototypes & Artifact Evaluations

For all 12 previously outlined works of this cumulative habilitation treatise, the respective hardware and software prototypes are publicly available under open-source licenses. The following enumeration provides the links to the source-code repositories of the respective publications.

1. *WoCA* [1]: Worst-Case Analysis of Device-Driven Intermittent Systems
   https://gitos.rrze.fau.de/woca
2. *ESP32-C3-OpenMAC* [2]: Open-Source Wi-Fi Drivers for ESP32-C3
   https://git.nt.uni-saarland.de/open-access/pfip
3. *Platin* [3]: Worst-Case Analysis Toolkit
   https://github.com/t-crest/platin
4. *TinyEP* [4]: TinyML-Based Runtime Energy Profiling
   https://zenodo.org/records/15193533
5. *FRET* [5]: Fuzzing-Based Dynamic Response-Time Analysis Tool
   https://git.cs.tu-dortmund.de/SYS-OSS/FRET
6. *FusionClock* [6]: Resource-Optimal Clock-Tree Reconfigurations
   https://gitlab.cs.fau.de/fusionclock
7. *Crêpe* [7]: Clock-Aware Preemption Control
   https://gitos.rrze.fau.de/crepe
8. *vNV-Heap* [8]: Non-Volatile Heap for Embedded Systems
   https://gitos.rrze.fau.de/i4/openaccess/vnv-heap
9. *PfIP* [9]: Power-Failure–Resilient Communication UDP/IP Stack
   https://git.nt.uni-saarland.de/open-access/pfip
10. *TinyBFT* [10]: BFT Framework for Constrained Devices
    https://gitos.rrze.fau.de/tinybft
11. *WatwaOS* [11]: Framework/OS for Resource-Aware Tailoring
    https://gitos.rrze.fau.de/i4/openaccess/watwaos
12. *CO_2CoDe* [12]: Carbon-Aware Co-Design Hardware Prototype
    https://gitos.rrze.fau.de/co2code

Artifact Evaluations Besides ensuring public availability, a focus of my research is on accounting for the FAIR principles [79] and on enhancing the reproducibility of research results. Specifically, whenever peer-reviewed artifact evaluations were offered by the venues where the respective publication appeared, the corresponding prototypes were submitted for undergoing the evaluation process. All artifact-evaluation submissions have been successful, and the related evaluations of *FRET*, *FusionClock*, *Crêpe*, *vNV-Heap*, *TinyBFT*, and *WatwaOS* are publicly available:

1. *FRET* [5] https://git.cs.tu-dortmund.de/SYS-OSS/FRET/
2. *FusionClock* [6] DOI-indexed artifact evaluation [15]
3. *Crêpe* [7] DOI-indexed artifact evaluation [14]
4. *vNV-Heap* [8] DOI-indexed artifact evaluation [13]
5. *TinyBFT* [10] https://gitos.rrze.fau.de/tinybft
6. *WatwaOS* [11] https://gitos.rrze.fau.de/i4/openaccess/watwaos

1.5 Subject Areas & Structure of Habilitation Treatise

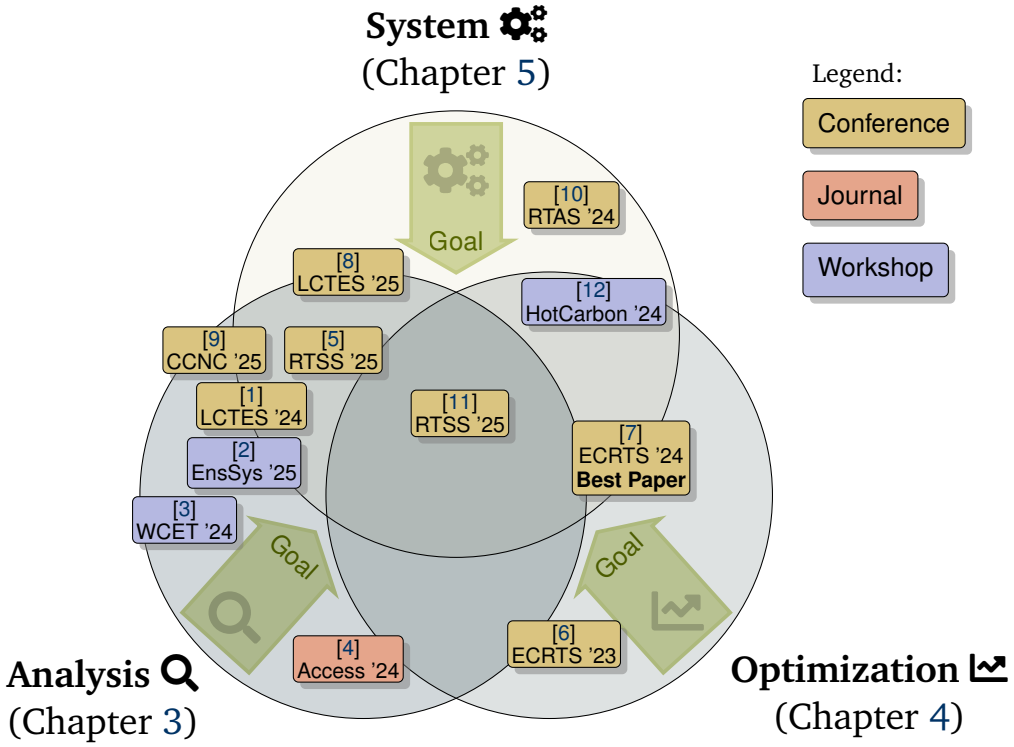

Figure 1.2: This treatise is concerned with the reliable and/or resource-optimized operation of embedded systems. For this operation, both analysis and optimization techniques constitute the essential foundation.

The core part of this habilitation treatise consists of the three subject areas of analysis, optimization, and systems themselves, which are illustrated in Figure 1.2. When papers consider two or more of these three areas, they are presented in the respective chapter being the most relevant for the contiguous presentation of this treatise. In this line of thought, the treatise's structure makes a differentiation between *system-aware analyses* and *analysis-aware systems*. Likewise, *system-aware optimizations* and *optimization-aware systems* are covered. Several examples of these differentiations are outlined subsequently: *WoCA* presents a system-aware analysis technique (based on device graphs) that incorporates specific device states of the overarching system and thereby reduces analysis pessimism. Although this research primarily targets the topic of systems (specifically, systems that heavily rely on power-consuming devices), the central contribution lies in the enhancement of the analysis technique. In contrast, the work on *vNV-Heap* [8] describes a library, serving as a runtime system, which is analysis-aware. It is designed in a way that enables determining upper bounds on the resource demand at any point during runtime under consideration of arbitrary input sequences (i.e., memory de-/allocations). The work on *WatwaOS* serves as an example for the optimization-related subject area, where the

resource-analysis and -optimizations techniques are aware of the target systems (covering both application and hardware). The *WatwaOS* analysis/optimization framework eventually generates a specialized operating system. Further, *WatwaOS*'s system-level abstractions are designed with the objective of enhancing the accuracy of worst-case analysis. The green arrows emphasize the eventual goals of having approaches with a high overlap of the three topics. System-aware analyses and optimizations are part of Chapter 3 and Chapter 4, respectively. The analysis-aware systems are the subject of Chapter 5. In detail, the treatise has the following structure:

Chapter 2 ("Background on Execution Under Resource Bounds") gives background information on the topics of (worst-case) program analysis and optimization techniques for the subsequent chapters. Further, it introduces fundamental terminology and explains the notion of tradeoffs and optimality.

Chapter 3 ("Static & Dynamic Program Analysis") outlines results of static & dynamic program-code analysis techniques.

Chapter 4 ("Optimizing Resource Demands") gives insights into two approaches that achieve energy reduction while maintaining runtime guarantees.

Chapter 5 ("Utilizing Analyses & Optimizations for Systems") presents several systems, such as an operating system or a runtime library, that make use of the previously discussed analysis and optimization techniques.

Chapter 6 ("Outlook & Conclusion") summarizes the results, gives an outlook on future works, and concludes this habilitation treatise.

Appendix A ("Bibliography") provides the treatise-related bibliography along with my personal bibliography.

Appendix B ("Paper Reprints") contains reprints of the 12 treatise-related, peer-reviewed papers, representing the core of this cumulative treatise.

CHAPTER 2

BACKGROUND ON EXECUTION UNDER RESOURCE BOUNDS

This chapter gives fundamental knowledge for the following chapters on analyses, optimizations, and their integration into systems. Section 2.1 gives an introduction to execution under resource limits. Section 2.2 introduces the terminology used. The notion of tradeoffs and optimality is given in Section 2.3.

2.1 Safe & Unsafe Execution Under Resource Bounds

System States & Critical Regions Software-controlled systems execute program paths that have distinct states over time. As an illustrative example, the value range of a variable in memory can have various states along a specific program path. Likewise, the memory demand itself of a system can be understood as a state varying over time. A general idea of evolving states is illustrated in Figure 2.1 by showing differently colored program paths with their states and transitions, inspired by Cousot's approach [80] to describing abstract

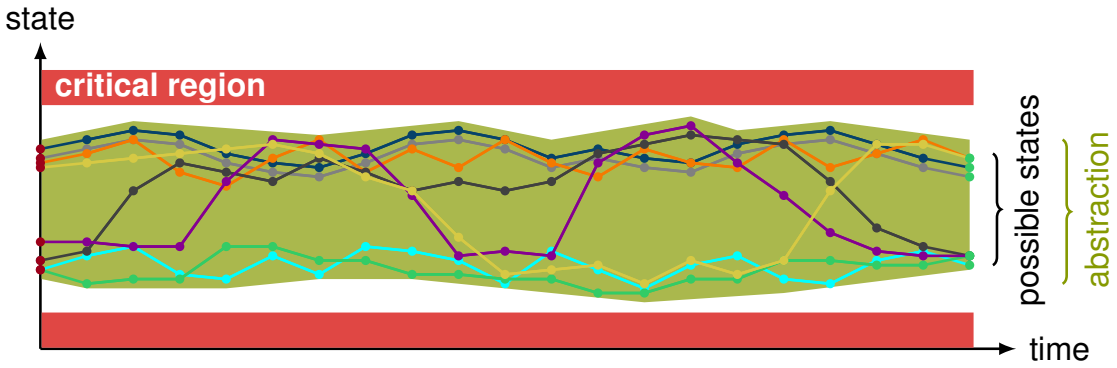

Figure 2.1: Capturing the concrete execution path with encapsulating abstractions (illustrated by the green area) is the fundamental idea of sound program-code analysis.

interpretation [81]. All of these program paths outline the concrete semantics of the system in general. In this figure, the two critical regions indicate possible states where the system enters a critical state: In view of memory resources, this state could mean overshooting the system's amount of validly accessible memory. With a focus on timeliness in real-time systems, entering a critical region with a program path could be considered as violating timing constraints and thereby missing deadline requirements. Similar considerations hold for energy-constrained systems where exceeding a given energy budget could cause the system to enter a state with inconsistent data or even to crash.

Abstracting from Concrete States For providing runtime guarantees to systems, general statements about all execution paths are necessary to guarantee that the system never enters a critical region. Consequently, runtime guarantees require a sound approximation for concrete program paths, as part of a static analysis approach. With regard to Figure 2.1, the green area visualizes the over-approximating semantics of the concrete program paths. While an unsound approximation, for example, determined through dynamic analysis, only covers a subset of all program paths, this illustration soundly and accurately encapsulates all states and transitions. The core idea of the majority of the presented works in this habilitation treatise (more specifically the works [1, 2, 3, 6, 7, 8, 9, 11]) is to abstract from concrete system states and to formulate a generic description of the system's behavior. Besides the works on abstracting from concrete semantics, two works [4, 5] specifically address dynamic analysis methods that usually underestimate the maximum resource demand. However, for actually giving guarantees on the safe execution within time budgets, a bound of a specific task's worst-case execution time (WCET) is necessary, which can be determined with abstract semantics [82]. Likewise, for the resource of energy, knowledge of the worst-case energy consumption (WCEC) [83] is crucial to guarantee the uninterrupted execution of operations within energy limits. To give one example, Chapter 3 covers the work on *WoCA* [1] in which WCEC bounds, determined through a static program-code analysis, enable forward-progress guarantees under an intermittent power supply. While the majority of this treatise's summarized works target soundness, the work on *FRET* [5], for instance, has the objective of a dynamic and, as a consequence thereof, inherently unsound analysis technique. Such dynamic techniques can be the only resort on modern hardware platforms where statically capturing the complexity of hardware behavior is impractical.

Formulating Worst-Case Path Problems with the Implicit Path-Enumeration Technique For determining resource bounds, explicitly enumerating all possible program paths and, thereby, comprehensively capturing the concrete program semantics is theoretically possible. However, from a practical point of view, this approach is infeasible with regard to the myriad of possible states in computer systems. To circumvent this problem of state explosion, the implicit path-enumeration technique (IPET) has been introduced [84, 85, 86], which is a widely used technique for solving the path-related problem of worst-case analysis. The core idea of the IPET is to formulate an integer linear program (ILP) problem. This problem can, in turn, be solved with mathematical optimization solvers (e.g., lp_solve [87], Gurobi [88],

CPLEX [89]). This type of ILP-based analysis works both for the WCET as well as for determining worst-case energy-consumption estimates [83]. In the ILP, the WCET analysis uses the temporal cost of blocks, while for the WCEC, the energy-related counterparts are used. For these two types of analyses, the IPET requires constraints; otherwise, for example, unbounded loops would lead to unbounded results. These constraints are derived from the program's execution paths. Regarding time, the IPET works for the intra-task [84, 85, 86] and the inter-task objective [26]. In analogy to time, the IPET also works for tasks in isolation [83] and for system-wide considerations [23].

Soundness Runtime guarantees are only possible with sound abstractions, which require both sound path analyses and models of the hardware. For the resource of time, such hardware models can be available for embedded platforms (e.g., for an ARM Cortex-M4 core [90]). However, with regard to the energy-related counterpart, such cycle-accurate models are not available in the manufacturers' documentation. To solve this problem, several approaches in this treatise model the energy demand by multiplying the target's maximum power demand P_{max} in a distinct power state by the respective WCET. Distinct power states are usually characterized by a fixed set of active devices and a fixed configuration of the subsystem. While this approach leads to an upper bound on the WCEC, it, at first, only shifts the problem of soundness because it relies on the availability of a bound on the maximum power P_{max} demand within distinct states. Fortunately, due to the coarser granularity of P_{max}, such (temperature-dependent) estimates are usually more often available in the documentation [91, 92, 93, 94]. Further, sophisticated approaches can achieve more accurate maximum power-demand estimates [95]. This treatise refers to upper bounds on the respective resource demand when a sound analysis has been applied. Further terminology is outlined in the following section.

2.2 Terminology

Observations & Bounds Regarding the resource of time, the term WCET refers to a task in isolation, without the interference of other (higher-priority) tasks or asynchronously occurring interrupts. The worst-case response time (WCRT) accounts for such disturbances and, thereby, denotes the end-to-end execution time. Specifically, the WCRT has a system-wide/whole-system perspective and, likewise, accounts for inter-task dependencies and preemptions through asynchronously occurring interrupts. In analogy to this terminology, but for the energy demand, the terms WCEC and worst-case response energy consumption (WCRE) are utilized. For test-based (i.e., dynamic) methods, where general statements on all possible program paths are not possible, and the observations refer to a subset of the program's semantics, the terms worst-observed execution time (WOET) and worst-observed response time (WORT) are used for timing properties. For the sake of completeness, analogous considerations hold for the respective energy-related terminology with worst-observed energy consumption (WOEC) and worst-observed response energy consumption (WORE). The terms that refer to worst-case estimates refer to upper bounds on the resource demand.

Treating Carbon as a Resource This treatise utilizes the term carbon to describe the *embodied* carbon footprint of embedded systems. More precisely, the metric embodied carbon describes an *equivalent* of carbon footprint that has been emitted along specific parts of the life cycle or the entire life cycle of an embedded system. At first glance, carbon does not appear to be a direct input to a system, which can be used for computation, as the resources of time, energy, and memory. Instead, carbon can be seen as an output *emission*: With regard to the standardized methodology for life cycle assessment (LCA) (under ISO 14040/14044), the global warming potential (GWP) (expressed in $kgCO_2eq$ as carbon-footprint equivalent) is one impact category [96]. Categories like freshwater use, human toxicity, or ecotoxicity comprise further impact categories. However, this treatise later argues (in Section 5.3) that embedded systems should be designed in a way that carbon is handled as a first-class resource. As a consequence, (embodied) carbon can be considered as an input resource for embedded systems design. This perspective of handling carbon as a resource for carbon-constrained systems is in accordance with time-, energy-, and memory-constrained systems in this treatise. The scarce available resources likewise pose a constraint for the system's design.

Specialization & Tailoring Versus Co-Design Since embedded systems are concerned about reducing resource demand, this treatise uses the term tailoring as a synonym for both specialization and customization. For example, the works [6, 7, 11] assume a given hardware platform, which offers a large configuration space in terms of performance and energy demand (more details on this configuration space are introduced in Section 4.2). That is, the available hardware resources are tailored to meet the resource constraints and functionality requirements of the application. For example, specialized clock configurations are selectively de-/activated on the target system to execute distinct sequences of the application. While specialization indicates one direction of an adapting infrastructure towards an objective (i.e., hardware $\Rightarrow$ software), the notion of co-design targets both directions of hardware $\Leftrightarrow$ software. In this context, the work on *CO_2CoDe* [12] targets system-level co-design for the objective of reducing the carbon footprint of embedded systems. The proposed approach envisions both a tailoring of hardware as well as software components.

2.3 Tradeoff & Optimality

Tradeoffs describe the relationship between objectives, as illustrated in Figure 2.2. The shown data points show both the WCRT over the WCRE of one task in a system with several tasks and interrupts under different configurations, analyzed with the *WatwaOS* framework [11]. Smaller values indicate better solutions since less resource demand is required in the related system configuration. More insights into how to achieve these configurations in the time-energy tradeoff are given in Section 4.2. All possible configurations are shown with the $\times$ markers. The symbols with the asterisks $\ast$ highlight solutions that are not dominated by any other solution. This property of dominance states that the respective configuration cannot further be improved for one objective (e.g., time) without

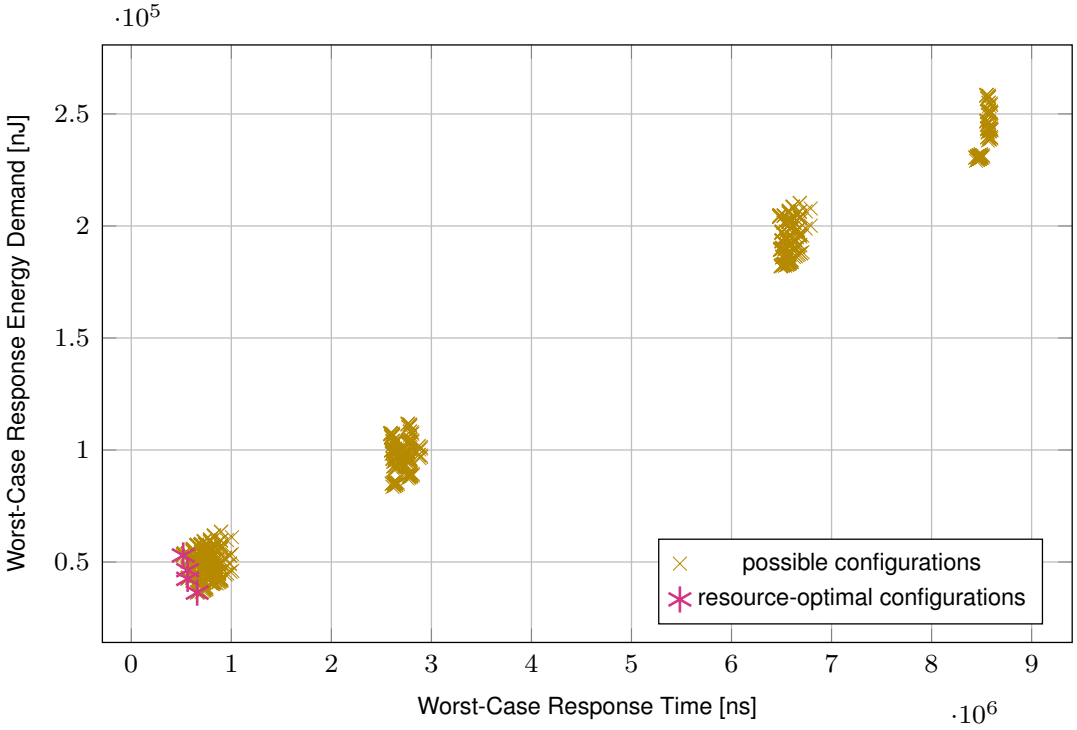

Figure 2.2: Required worst-case energy and runtime resources of executed tasks, analyzed with *WatwaOS* [11]: In this setting, the clock tree offers numerous configuration options. Within the possible configuration space offered by the available hardware, the challenge is to best exploit the possible tradeoff and to eventually determine worst-case–optimal configurations.

worsening the other objective (e.g., energy). The *Pareto frontier* highlights these points that are not dominated by any other system configuration. In Figure 2.2, four configurations characterize this Pareto frontier. These four points are considered to be the *optimal* solutions in this scenario. However, this property of optimality only holds when neither the given execution-time budget nor the energy-consumption budget is exceeded by the respective solution. Because each configuration refers to both a WCRE and WCRT estimate, which bound the respective worst-case resource demands, the Pareto frontier highlights the system's *worst-case–optimal solutions*.

CHAPTER 3
STATIC & DYNAMIC PROGRAM ANALYSIS

The subsequently outlined analysis techniques form a foundation for subsequent optimization techniques, which eventually make systems more reliable and/or more resource-efficient.

3.1 Program Analysis for Runtime Guarantees

Several standards from industry, such as ISO 26262 (automotive), DO-178C (aerospace), IEC 62304 (medical), and EN 50128 (railway) give explicit recommendations to use static analysis techniques in order to comply with the respective standards. As outlined in Section 2.1, in contrast to testing, static analysis techniques are the only means to provide provable runtime guarantees. With regard to finding bugs, the return on investment of using static analysis techniques for mission-critical software at NASA has been explored [97], concluding with the clear benefits of adopting static analysis in a product's life cycle. To verify timeliness, the use of static WCET techniques is, for example, practically beneficial for critical avionics, automotive software projects, satellites, or the energy industry [98, 99]. For the domain of timeliness, numerous WCET tools [98, 100, 101, 102, 103, 104, 105, 106, 107, 108] have been introduced. Likewise, for determining WCEC bounds, several analysis approaches exist [23, 28, 83, 109, 110]. This habilitation treatise has a focus on describing both timing- and energy-related analyses for tasks and their interaction with the OS and with OS-controlled devices. In order to conduct such analyses, suitable abstractions are crucial, which are outlined in the subsequent section.

3.2 Abstractions For Whole-System Analyses

Atomic Basic Blocks Atomic basic blocks (ABBs) are fundamental system-level abstractions, which serve for several works of the outlined papers [1, 5, 6, 7, 11]. The original work on ABBs has been initiated at Friedrich-Alexander-Universität Erlangen-Nürn-

berg (FAU) during the course of Fabian Scheler's dissertation in 2011 [111] and his preceding works [112, 113, 114]. His introduced research contribution on the *Real-Time Systems Compiler (RTSC)* [115] fundamentally relies on the abstraction of ABBs. The work on ABBs has not been the first approach to introduce compiler techniques for improving the analysis of embedded real-time systems [116, 117, 118]. However, the strong connection of the line of work on ABBs with software standards for real-time systems, specifically OSEK [119], supported further system-level research [120, 121, 122, 123, 124]. An ABB is a superstructure of a single or multiple control-flow graphs (CFGs). While a CFG connects basic blocks (BBs), an ABB graph relates multiple connected ABBs, which, in turn, can subsume one or more BBs. An ABB is generally terminated by a system call (or syscall for short), which transfers the control flow into the OS kernel. From the perspective of a scheduler, being located inside the OS, ABBs are dispatched atomically, hence the name. The initial abstraction of ABBs targeted real-time systems and their semantics. For the context of energy-aware systems, the abstraction has been later extended to address changes in the power-related states, which resulted in the power atomic basic blocks (PABBs) [23]. The properties of a PABB are that (1) it inherits all properties of an ABB and (2) it has a common set of active devices, which is essential for determining energy-consumption bounds of code in embedded systems.

State Transition Graphs While the ABB as well as PABB graphs do not carry any context-specific (i.e., program-path–specific) knowledge, the related data structures of the STG or the power-state–transition graph (PSTG) contain all possible program paths along with path-sensitive information. These representations are constructed by an explicit path enumeration of the respective ABB/PABB graphs. While explicit path enumerations are infeasible for CFGs and their BBs due to the impractically large search space, an explicit enumeration practically works on ABB/PABB graphs because of the coarser granularity and the, as a consequence thereof, smaller search space. Further, code for embedded systems usually has known and bounded structures that make explicit enumerations computationally feasible in comparison to application code that serves more generic purposes. The ABB/PABB representations constitute central conceptual foundations for several works in this habilitation treatise [1, 5, 6, 7, 11]: For example, the work on *WatwaOS* [11] builds upon the notion of PSTGs and extends it with node-merging techniques with the purpose of finding worst-case–optimal solutions in the time-energy tradeoff. Further, the work on *WoCA* introduces device graphs to the system-related PSTG for the application scenario of intermittent computing, which is outlined as follows.

3.3 Benefits of Worst-Case Analyses for Intermittent Computing

Intermittent Computing & Checkpointing The Internet of Batteryless Things [68] comes with the challenge of surviving blackout phases during computing. Due to the unreliable power sources (e.g., solar cells), the systems face intermittent execution and have to checkpoint relevant state in non-volatile memory across off-phases. For checkpointing, the hardware platforms from the MSP430FR family, being widely used for intermittent

computing, have NVM directly mapped to the RAM. By considering the registers in the checkpoint, such platforms support incremental progress. However, devices usually have different semantics and require larger transactions to safely complete: For example, sending out packets requires the related transaction to complete within given energy-budget bounds and, as a consequence thereof, without power failures. Subsequently, the use of worst-case analysis techniques is outlined, which can give guarantees on the completion of such transactions.

Safe Execution Under Energy Budgets The worst-case resource-consumption analysis techniques originate from verifying timeliness in real-time systems [82]. Later, they were adopted for determining WCEC estimates [83]. In analogy to ensuring that tasks meet their deadline and execute within the given time budget by using their WCET estimate, the energy-related counterparts are able to guarantee that tasks execute within given energy budgets. Consequently, the use of worst-case energy-consumption estimates can be useful for the domain of intermittent computing [125, 126]. Prior, for energy-constrained systems, static WCEC estimates were used to guarantee safely persisting states and waking up from them in order to conduct useful work [27, 33].

Accounting for Devices The previously outlined problem of transactional semantics and the fact that devices temporarily demand most of the system's drawn power puts devices into the spotlight of worst-case analyses in the context of intermittent computing. For example, considering a common embedded SoC platform (i.e., ESP32-C3 [127]), the power demand can range from a few $\mu\mathrm{W}$ to several hundred mW with active power-consuming components. Modeling the power behavior of single instructions is possible [109, 128, 129, 130] and already involves a degree of pessimism. However, the variances in the power demand across individual instructions, when executing with a distinct processor configuration, can end up being negligible in relation to the orders of magnitude that originate from changing power configurations. Consequently, the power-related behavior of devices and their respective configurations should be as accurate as possible to yield bounds with an acceptable degree of pessimism. *WoCA* [1] introduces the notion of *device graphs* into the analysis of WCRE bounds, also covering the influence of asynchronously occurring interrupts. While such graphs have been previously defined, for example, for transceivers [131, 132, 133], *WoCA* introduced their use for context-sensitive path exploration to accurately determine energy-demand bounds. Specifically, the context-sensitive information on device states is part of the system-state exploration, which, in turn, serves for determining an ILP formulation. Solving this with optimization solvers eventually yields WCRE estimates. The infrastructure of *WoCA*, with its device graphs focusing on long range (LoRa) communication, later supported the work on *PfIP* [9] and its utilized Wi-Fi drivers [2].

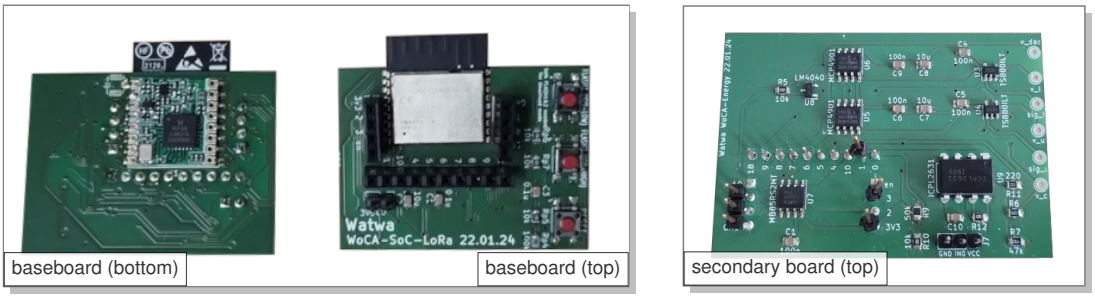

Figure 3.1: Providing runtime guarantees for intermittent systems requires both soft- and hardware support: Before an impending power failure, a hardware circuit notifies the runtime system to enter a persistent state. *WoCA* has, besides its main computing platform serving as a baseboard (left side), a shield (right side) that monitors the state of charge and releases interrupts when configurable thresholds are met.

3.4 Hardware/Software Integration in Intermittent Systems

While approaches for intermittent computing exist that do not rely on hardware support for energy management, the outlined work on *WoCA* utilizes a dedicated hardware platform, which is a requirement for just-in-time checkpointing approaches [134, 135, 136, 137]. This platform, illustrated in Figure 3.1, has the purpose of monitoring the available energy without the requirement of synchronously polling the state: The combination of a digital-to-analog converters (DACs) connected to comparators offers the possibility to configure voltage thresholds at which asynchronous interrupts are released when the capacitor's voltage reaches this threshold.

Availability of Hardware Behavior & Device Drivers As the evaluations of *WoCA* validate, the availability of worst-case estimates has the benefit of avoiding starvation compared to approaches that are unaware of the transactional behavior of devices. However, providing such guarantees requires in-depth knowledge of the hardware's behavior. For the LoRa device, as shown on the bottom side of the baseboard in Figure 3.1, the detailed behavior is available [92]. Unfortunately, information with a sufficient level of detail or open-source drivers is not always available. The UDP/IP stack with transactional semantics *PfIP*, later outlined in Section 5.1, uses Wi-Fi for communication on the ESP32-C3 platform. To enable Wi-Fi communication on the popular Xtensa-based ESP32 SoC, the *OpenMAC* [69, 70] project reverse-engineered the Wi-Fi stack to have an open-source MAC layer. As part of the overarching work on *PfIP*, the work on *ESP32-C3-OpenMAC* [2] introduced open-source drivers for the RISC-V–based ESP32-C3 SoC, which eventually allowed Wi-Fi–enabled systems to be statically analyzed.

3.5 Static Analysis Framework *Platin*

All main works of this habilitation treatise outlined so far in this chapter [1, 2] use the framework *Platin* for the static analysis. Likewise, further works presented in subsequent

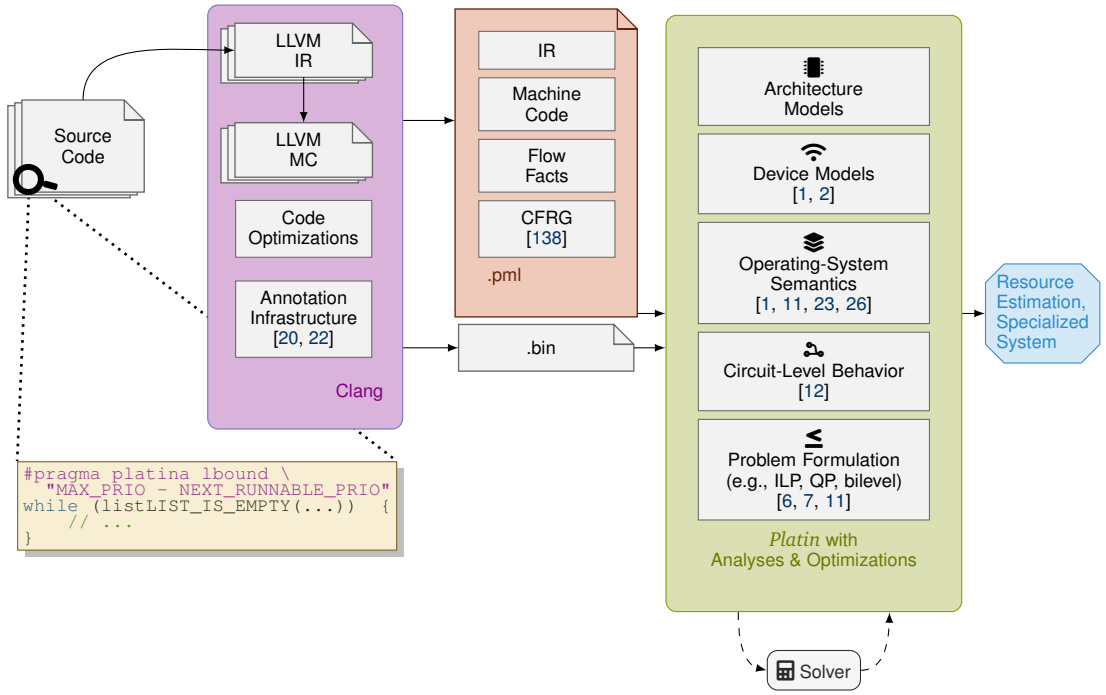

Figure 3.2: Overview of the *Platin* framework and the surrounding Clang/LLVM compilation infrastructure, along with the associated system-level analysis and resource-optimization techniques.

chapters (i.e., *FusionClock* [6], *Crêpe* [7], *PfIP* [9], *WatwaOS* [11]) rely on this tooling infrastructure. *Platin* has been first presented as part of the T-Crest project [139, 140]. The name *Platin* originates from its objective of being a portable LLVM annotation and timing toolkit. As the name suggests, *Platin* is structured around the LLVM compiler infrastructure [141]. The overall framework is visualized in Figure 3.2. *Platin* relies on an extended version of the Clang compiler frontend, enabling the support of the programming languages C and C++. Based on the results of a Master's thesis [142], preliminary efforts exist to integrate Rust as a further supported language. This line of work is motivated by the tendency to use memory-safe programming languages [143, 144] for critical applications without sacrificing performance. The work on *Platin* [3] summarizes enhancements that have been introduced over a period of more than ten years. These enhancements cover, for example, the support of analyzing whole real-time systems with regard to the WCRT [26] and their WCRE [23]. In these works, the analyses consider multiple layers in the system's stack (🗐) that span from the hardware, through the OS (with its scheduling semantics and its possibility to serve asynchronous interrupts), to the application with fixed-priority tasks. Further, the infrastructure supports annotations [20, 22] to formulate flow facts of the system's behavior. With this annotation infrastructure, incorporating parametric knowledge of the system into the analysis (i.e., the length of the scheduler's ready queue) is possible. Information in the *Platin* framework is mainly forwarded with *Platin*'s Program Metainfo Language (PML) format. The *control-flow–relation graphs* (CFRGs) enable tracking flow

facts and lowering these facts from LLVM intermediate representation (LLVM IR) down to the lower abstraction level of machine code [138]. With regard to *Platin*'s internals, visualized in the green box in Figure 3.2, it supports several architecture models (), with the Patmos architecture [145, 146] having the most microarchitecture-aware integration into *Platin*. Further basic infrastructure for ARM Cortex-M4 [36] (Infineon's XMC4500), RISC-V [6] (ESP32-C3 with bypassing caches), and AVR is available. Device models () can address devices for analyzing LoRa-based [1] or Wi-Fi–based [2] communication. For whole-system analysis, the OS and scheduling semantics are part of the analysis [11, 23, 26]. Accounting for effects on the circuit level () is necessary to address effects that propagate, for example, from the energy storage through the embedded system's entire stack [12]. Eventually, the infrastructure features the formulation of problems (), such as ILPs, quadratic programs (QPs), or bilevel problems, which are solved by external mathematical optimization solvers (), for example, lp_solve [87] or Gurobi [88].

Modeling Device Accesses The following line of source code illustrates the challenge of modeling the behavior of memory-mapped device accesses:

```
*((volatile uint32_t*)0x60033ca0) &= 0xff00efff
```

While the microarchitectural modeling () accounts for the time/energy behavior of instructions, it has no awareness of possible memory-mapped interactions with devices. In this specific example on the ESP32-C3, the write access to memory, after first reading the value and computing the result, requires several cycles of execution time on the CPU. However, from the perspective of the energy demand and the power-related behavior, this operation activates the SoC's internal Wi-Fi transceiver and sets it from sleep state to standby mode. All instructions succeeding this activation would require a higher energy demand due to the SoC's overall higher power demand. The device models (), integrated into *Platin*, have to account for these resource-consumption semantics and the hardware context along program paths for accurate analysis.

3.6 Dynamic Analysis

As introduced in Section 2.1 in this treatise's fundamentals, static program code analysis techniques that utilize (1) a sound program-path analysis technique combined with (2) a sound hardware model are able to yield sound resource limits, which are, in turn, necessary to give proven runtime guarantees. Unfortunately, developing sound hardware models is increasingly becoming impractical. With regard to timing, the complexity of modern hardware hinders the development of sound models, which is subsequently detailed.

Problem of Complex COTS Hardware & Timing Certification First, sophisticated hardware platforms make it impossible, from a practical point of view, to determine sound hardware models given the high complexity of the hardware. Second, commercial-off-the-shelf (COTS) hardware platforms are becoming increasingly popular for developing embedded

systems, also for the use of safety-critical scenarios. For these COTS platforms, specific microarchitectural details, which are required for deriving sound hardware models, may be undisclosed by manufacturers [147]. To support the use of COTS but still provide the best possible timing estimates, dynamic test-based methods are also considered for allowing certification in highly safety-critical environments [148].

Problem of Circuit-Level Energy Transparency The problem of undocumented hardware details becomes more challenging when considering the resource of energy: The system's eventual power demand depends on the actual circuit level and might even differ between circuit boards due to manufacturing process variations [149]. Combining the machine code and microarchitectural level is sufficient for timing analysis. Unfortunately, for energy-related analyses, these considerations are no longer sufficient, since they require the analysis to include layers below, namely, the *circuit level*. For instance, the energy supply has an impact on the energetic behavior of the system, as further detailed in *CO_2CoDe* [12]. Nevertheless, even in the presence of well-documented SoC platforms with regard to energy, the granularity is more coarse-grained compared to cycle-accurate timing. Finally, while the timing behavior is usually very stable in various environmental conditions with varying ambient temperatures, the energy demand can be highly temperature dependent: A temperature increase of $100\,^{\circ}\mathrm{C}$ can ramp up in power demand by a factor of $3.5\times$ [149]. For the targeted embedded system considered in the *TinyEP* approach, the observed power variances were smaller, however, significant: The observed average power demand varied between $132\,\mathrm{mW}$ at $-40\,^{\circ}\mathrm{C}$, over $138\,\mathrm{mW}$ at $21\,^{\circ}\mathrm{C}$, up to $152\,\mathrm{mW}$ at $80\,^{\circ}\mathrm{C}$. When static analysis techniques are not feasible, for example, to precisely account for such environment-induced variances, dynamic techniques are a viable substitute to yield practical, however—with regard to the actual worst case—under-approximating resource-demand estimates. In the case of *FRET*, the dynamic analysis technique determines estimates ahead of runtime, while the *TinyEP* approach acts during the system's runtime, which is outlined in the subsequent section.

3.6.1 Dynamic Energy Assessment Under Varying Environmental Conditions

The currently ongoing trend of using artificial intelligence comes with sophisticated frameworks for running machine-learning applications even on highly resource-constrained devices. This research field is often referred to as *TinyML*, with TensorFlow Lite Micro (TFLM) being one widely used framework for on-device inference [71, 72]. With regard to the problem of online energy-demand assessment under varying environmental conditions, *TinyEP* [4] introduced an approach to make online energy predictions of various tasksets.

Combining ADCs & Coulomb Counting For online energy predictions, a common approach is to use analog-to-digital converters (ADCs) for sampling the voltage over a shunt resistor R_{shunt} and compute the power demand according to $P = \frac{V_{\text{shunt}}}{R_{\text{shunt}}} \cdot V_{\text{system}}$, where the system's voltage V_{system} is assumed to be constant (e.g., $3.3\,\mathrm{V}$). However, sampling the system with

a high frequency comes with the downside of having a high overhead, which is especially undesired in systems with scarce available resources. An approach that introduces less overhead is the utilization of coulomb-counting circuits, also marketed as fuel-gauge integrated circuits (ICs). These circuits also have the benefit of less suffering from the concept drift introduced by changing environmental parameters (i.e., temperature, humidity) compared to ADCs. However, these circuits have substantially higher sampling rates compared to ADCs and are, consequently, not usable for tracking the energy demand of single task executions. To solve the problems of low-overhead energy predictions under varying environmental conditions, *TinyEP* uses a TinyML model that learned relevant patterns of the system's power behavior based on a previously executed training set. This leads to the benefit of reducing the sampling rate of the on-board ADC, which, in turn, reduces the sampling-related overheads. To compensate for concept drift, *TinyEP* utilizes a coulomb-counting circuit that occasionally gives a ground truth. The evaluations of *TinyEP* have shown that the framework can accurately estimate power demands under high temperature ranges from $-40\,^{\circ}\mathrm{C}$ up to $80\,^{\circ}\mathrm{C}$.

3.6.2 Dynamic Fuzzing-Based Response Time Analysis

Problem of Missing System-Level Semantics Since analysis techniques can be practically inapplicable to highly complex COTS hardware or might suffer from an unrealistically high degree of pessimism, dynamic approaches are a feasible approach to addressing resource-consumption estimates. End-to-end measurements are utilized in hybrid timing analysis approaches [105], where the measurements serve as input for an ILP-based static analysis. However, existing approaches miss the effects that originate from system-level interactions, such as synchronous task communication or asynchronously occurring interrupts.

Solution: Capturing System States for Driving Fuzzing Heuristics In the context of determining more accurate WORT estimates, the work on *FRET* [5] presents an approach that relies on the technique of fuzzing. Fuzzing has been introduced in the 1990s for testing software [150]. While it is (still) a very active research area [151] for detecting security flaws, this type of testing has not received attention for determining WORT estimates. *FRET* [5] approaches the problem of finding such estimates with awareness of possible system states, specifically with knowledge of a dynamically determined representation of the state-transition graph (see Section 3.2).

Main Results The coverage-guided fuzzing [152, 153], where the fuzzer tries to reveal new basic blocks, serves as one baseline for *FRET*'s evaluation, which is by default not aware of the semantics of OSes and dependencies between tasks. The main results of *FRET* show that the proposed fuzzing strategy based on the representation of an observed STG outperforms coverage-guided fuzzing. The novel fuzzing strategies for mutating inputs cover the combination of input snippets along paths that were observed along the constructed STG. Further, *FRET* injects input edges that represent interrupts where no asynchronous transitions have been observed before. The results include that *FRET* finds

larger WORT estimates faster and more reliably compared to the baseline approach. As further outlined in Section 6.2, an avenue of future work exists that uses *FRET* and combines both (1) the explored STG, which is usually a subset of the actual STG that does not include all possible transitions, and (2) the use of hybrid timing-analysis techniques [105].

3.7 Main Analysis-Related Papers

In view of this chapter's focus on analysis techniques for embedded systems, the following list shows the five related papers. The first three papers correspond to the static analysis approaches and the latter to dynamic analysis techniques.

LCTES '24 Phillip Raffeck, Johannes Maier, and Peter Wägemann [1]
WoCA: Avoiding Intermittent Execution in Embedded Systems by Worst-Case Analyses with Device States
Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '24)

ENSsys '25 Ishwar Mudraje, Jasper Devreker, Kai Vogelgesang, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, and Thorsten Herfet [2]
Reverse Engineering the ESP32-C3 Wi-Fi Drivers for Static Worst-Case Analysis of Intermittently-Powered Systems
Proceedings of the 13th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems (ENSsys '25)

WCET '24 Emad Jacob Maroun, Eva Dengler, Christian Dietrich, Stefan Hepp, Henriette Herzog, Benedikt Huber, Jens Knoop, Daniel Wiltsche-Prokesch, Peter Puschner, Phillip Raffeck, Martin Schoeberl, Simon Schuster, and Peter Wägemann [3]
The Platin Multi-Target Worst-Case Analysis Tool
Proceedings of the 22nd International Workshop on Worst-Case Execution Time Analysis (WCET '24)

Access '24 Kilian Müller, Johannes Weidner, Norman Franchi, and Peter Wägemann [4]
TinyEP: TinyML-enhanced Energy Profiling for Extreme Edge Devices
IEEE Access

RTSS '25 Alwin Berger, Simon Schuster, Peter Wägemann, and Peter Ulbrich [5]
Dynamic Fuzzing-Based Whole-System Timing Analysis
Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)

CHAPTER 4
OPTIMIZING RESOURCE DEMANDS

This chapter outlines optimization techniques for reducing resource demands in embedded real-time systems by exploiting time-energy tradeoffs. In essence, this chapter is based on two works that are central parts of the overarching DFG-funded research project *Whole-System Optimality Analysis and Tailoring of Worst-Case–Constrained Applications (Watwa)*, which is subsequently introduced (see Section 4.1). The respective works (*FusionClock* [6], *Crêpe* [7]) are put into *Watwa*'s context in Section 4.2 and listed in Section 4.3.

4.1 The *Watwa* Research Project

Motivation Example & Notion of Devices Figure 4.1 serves as an illustrative example to outline the core idea of the *Watwa* research project. The shown variants of the tasks have the same functional behavior: They consecutively read out a sensor and compute the arithmetic mean from these samples[1]. From a generic perspective, *any power-consuming and software-configurable component* is considered to be a *device* in this treatise. In this example, both the processor (assuming a single-core system) and the sensor device are mandatory devices to compute the result. While the processor always has to be active, the sensor can be selectively activated to save energy. This is especially relevant since input devices, like ADCs for sampling signals, can demand a significant amount of power. With regard to the variants, the first software variant (a) activates the sensor prior to entering the loop and deactivates it afterwards. Variant (b) chooses a more fine-grained approach and activates/deactivates the sensor right before/after calling the function to yield the sensor's value. The last variant (c) avoids the selective de-/activation and globally enables the sensor. The *Watwa* project approaches the question of the optimal solution with regard to the application's constraints in this example. For the determination of an optimal solution ahead of runtime, the worst-case resource-consumption behavior is considered in order

[1] For the sake of simplicity, range checks due to possible overflows are omitted in the example.

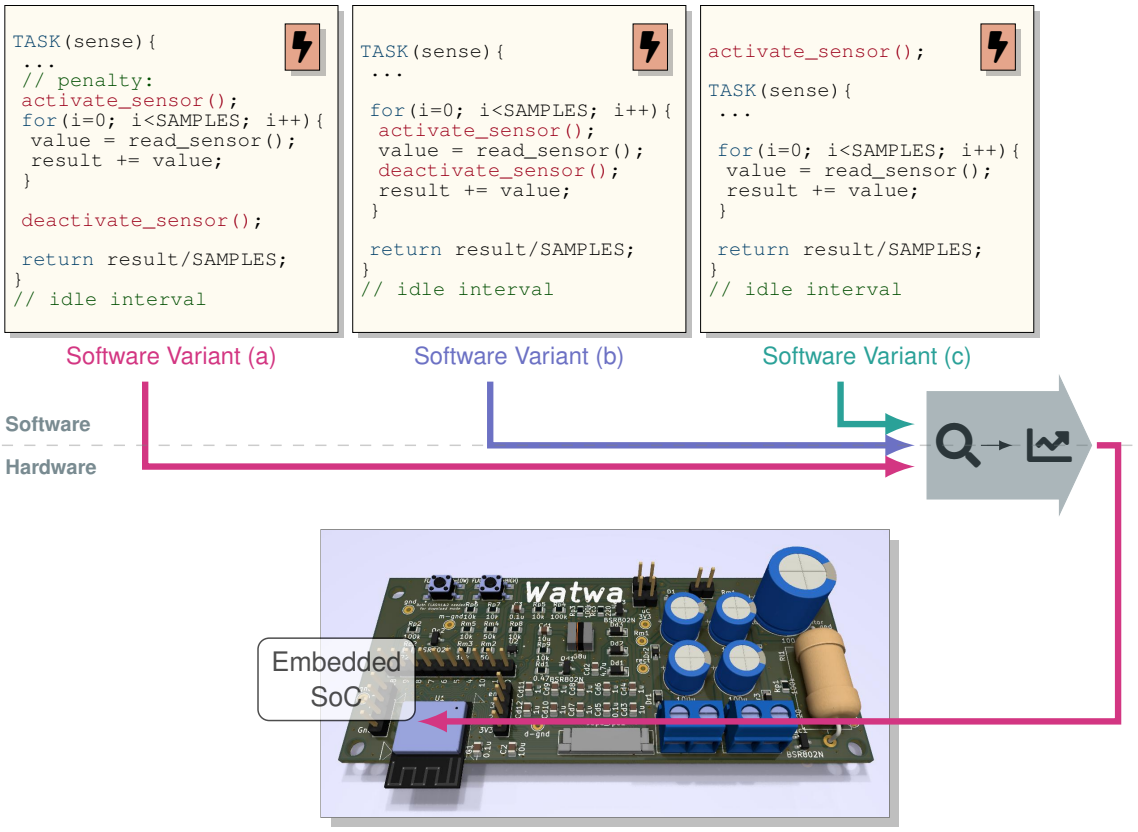

Figure 4.1: *Watwa*'s functionally equivalent variants of the application's code have different resource-consumption behaviors. The related search space comes with the challenge of finding resource-optimal solutions with regard to the application's constraints (e.g., precedence constraints with regard to active devices).

to achieve runtime guarantees. Consequently, the application is analyzed (as illustrated with the 🔍) concerning the worst-case behavior. Based on the analysis results, the best solution is selected in the optimization phase 📈. Although this indicates sequential steps, in fact, analysis and optimization can be interwoven in a common problem formulation (see Section 5.2).

Hard- & Software Constraints To accurately answer the question concerning the property of worst-case optimality, several constraints have to be respected in the search space. Switching hardware configurations, for example, to activate the sensor, involves a certain penalty, which leads to increased runtime and energy demand. When switching the hardware configuration of the processor itself, substantial penalties are likewise possible. Besides these hardware-related constraints, the application also comes with software-related constraints: For example, when calling `read_sensor()`, the device needs to be activated (once) as a precedence constraint. Shifting the call to `activate_sensor()`

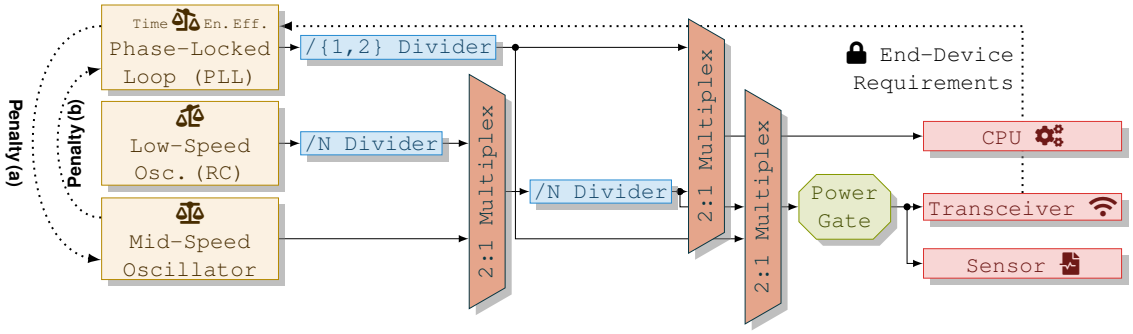

Figure 4.2: Complex clock subsystems with heterogeneous input clock sources offer a fine-grained selection in the time-energy tradeoff. The constraints that originate from end-device requirements need to be addressed as well as the penalties between arbitrary configurations.

offers space for optimizations. Within this space, the objective to find worst-case–optimal solutions needs to include these software constraints. Further constraints originate from the executed task set: Calling `TASK(sense)` with a short period might justify activating the sensor only once. Considering solely synchronous task activations usually yields a comparably small number of possible system states, under the assumption of applications with around a dozen tasks [23]. The problem of analyzing and optimizing whole systems becomes substantially more complex with asynchronous events, such as the presence of asynchronously occurring interrupts, as indicated by the ϟ symbol in Figure 4.1. To summarize, for finding optimal solutions (1) hardware-related penalties, (2) the hardware's device constraints (further explained in Section 4.2), (3) the application's device constraints, (4) the application's timing constraint (e.g., periodicity of tasks), and (5) the presence of asynchronous events have to be considered, which covers the scope of the *Watwa* project.

Origin of *Watwa* The main idea of having functionally equivalent task variants under the consideration of the application's requirements, such as the outlined device-related precedence constraints, came up during my doctoral thesis [58]. The outlook of the thesis motivated a combined consideration of time and energy for the formulation of optimization problems. Following the ongoing trend of having more heterogeneity and complexity, for example, due to the availability of multiple clock sources on embedded SoC platforms for system-level optimizations, has become a further ingredient of *Watwa*.

4.2 Heterogeneous Clock Subsystems in the Time-Energy Tradeoff

Spanning the Search Space Modern embedded SoC platforms offer a huge configuration space targeting performance and energy efficiency. To reconsider the mentioned SoC ESP32-C3 [127], the system can run with speeds up to 160 MHz and have power demands ranging from several hundred mW down to single-digit μW, when the system enters low-power sleep modes. The core control unit for configuring this time-energy tradeoff is the *clock subsystem*, also referred to as the clock (distribution) tree or clock

network. Figure 4.2 illustrates an excerpt of a representative subsystem from the ESP32-C3 SoC [127]. Comparable embedded hardware platforms feature such clock subsystems with similar tradeoffs and configuration spaces (e.g., STM32-based SoCs [154]). Most dividers, shown in blue in Figure 4.2, offer many configurable settings, each setting resulting in a multiplier for the general configuration space. Multiplexers and power gates either route or switch off the signal, with the latter reducing the system's static power demand. Further, the heterogeneity of the SoC comprises multiple clock sources: These input sources differ in their maximum possible speeds, their energy efficiency, their temperature stability, and their clock stability. Changing between the clock configurations comes with context-specific penalties. That is, switching from the phase-locked loop (PLL) involves **Penalty (a)** while switching to it demands **Penalty (b)**. The differences originate from technical details in the hardware: For example, reconfiguring or activating the PLL can demand a temporal penalty, because of the duration until the PLL is stabilized and thus in a locked state. On the output side, components integrated into the SoC come with device-related requirements: For example, the Wi-Fi transceiver (📶) can only run with a specific configuration of the PLL input source. This exemplary end-device requirement of an output device demanding a specific input-device configuration is visualized with the 🔒 symbol. These end-device requirements constitute restrictions on possible configurations in the overall search space.

Finding Solutions in the Search Space State-of-the-art approaches on energy-aware real-time systems mainly used a power model with a static P_s and a dynamic portion $P_d(f)$: $P(f) = P_s + P_d(f)$ [155]. The dynamic part takes the frequency f as the only variable for the power demand. However, this approach falls short with respect to modern clock subsystems. While the works of ScaleClock [76] and Power Clocks [77] put a focus on the relevance of these modern clock sources, they are not able to give runtime guarantees based on static program analysis. *FusionClock* [6] has approached this gap by employing an analysis technique for the application's tasks in the device-aware configurations. It demonstrates that the awareness of feasible configurations along with their related costs can be formulated as a minimum-cost flow problem, which can be solved with optimization solvers.

Controlling Asynchronous Preemptions The work on *FusionClock* considered strictly periodic task sets. While this system model holds for many systems, numerous real-world systems demand the handling of asynchronously occurring interrupts, for example, when data is available from devices that provide input data (e.g., sensors, transceivers). Based on the findings of *FusionClock*, the work on *Crêpe* [7] extended the original work and introduced a scheme for preemption control: After the execution of individual jobs, interrupts are synchronously considered by polling the interrupt source. *Crêpe* reacts to the presence of interrupts by changing to alternative schedule tables. Relying on schedule tables precomputed in memory, *Crêpe* exploits a further tradeoff besides the time-energy tradeoff, namely, the time-memory tradeoff: Storing alternative scheduling sequences requires a substantial amount of memory. However, when staying with the whole system's memory below the available limit, this tradeoff can be exploited to optimize the performance.

Main Results Results of *FusionClock* and *Crêpe* include that dynamic voltage and frequency scaling (DVFS) approaches, which have been extensively explored in the scope of real-time scheduling [155], are no longer directly applicable to systems with intertwined clock subsystems and end-device requirements. To solve this problem of heterogeneity, the outlined works make use of a model of the clock subsystem and combine it with device requirements in a quadratic program (QP) formulation. Under consideration of these clock subsystems and their supported time-energy tradeoff, the two works *FusionClock* and *Crêpe* are the first to determine worst-case–optimal schedules that provably minimize the energy demand while guaranteeing timeliness.

Towards the Goal of Integration into Systems The works on *FusionClock* and *Crêpe* have a focus on resource optimizations. Based on these results, the following chapter shows, with *WatwaOS*, the integration into a surrounding framework, which eventually generates an OS, tailored to the target application.

4.3 Main Optimization-Related Papers

With the main focus on optimizations under resource limits, the following two papers are part of the *Watwa* project and constitute the main papers of this chapter.

ECRTS '23 Eva Dengler, Phillip Raffeck, Simon Schuster, and Peter Wägemann [6]
FusionClock: Energy-Optimal Clock-Tree Reconfigurations for Energy-Constrained Real-Time Systems
Proceedings of the 35th Euromicro Conference on Real-Time Systems (ECRTS '23)

ECRTS '24 Eva Dengler and Peter Wägemann [7]
Crêpe: Clock-Reconfiguration–Aware Preemption Control in Real-Time Systems with Devices
Proceedings of the 36th Euromicro Conference on Real-Time Systems (ECRTS '24)

CHAPTER 5
UTILIZING ANALYSES & OPTIMIZATIONS FOR SYSTEMS

This chapter outlines approaches that utilize analyses and optimizations for improving the efficiency and/or the reliability of embedded systems. The works [8, 9, 10] address systems with power-failure resilience in Section 5.1 in the light of increasing functionality. Section 5.2 is centered around *WatwaOS* [11] and comprises worst-case–aware tailoring. While all previously outlined works targeted time, energy, and memory resources, Section 5.3 eventually broadens the scope with *CO_2CoDe* [12]. The section presents an avenue towards system-level co-design for optimizing embedded systems, with one objective being the reduction of carbon footprints. At the end of this chapter, Section 5.4 lists all systems-related papers of this cumulative habilitation treatise.

5.1 Increased Functionality of Systems with Power-Failure Resilience

Designing and implementing embedded systems inherently come with the contrasting objectives of (a) integrating more functionality while (b) making them smaller in terms of size, weight, and power. The subsequently discussed systems follow this trend of increasing functionality and supporting novel applications in the context of power-failure–resilient and analysis-aware embedded systems.

Making Systems Analysis-Aware Running embedded systems under power failures involves solving the challenge of persisting states across blackout periods. Before an impending power failure, the system must be checkpointed to non-volatile memory. The envisioned Internet of Batteryless Things comprises connected systems that face power failures due to the unstable power supply. As already outlined in Section 3.3, the use of worst-case analysis techniques can be beneficial to give guarantees on the forward progress of systems. However, the systems themselves also have to be designed in a way to allow analyzability.

Programming-Language & Compiler Support for Analysis-Aware Systems WCET analysis tools like `aiT` [98] start from the binary code and reconstruct the control flow from this code representation [156, 157]. Although this approach has the benefit of being agnostic to specific compilation toolchains, it has the drawback of reconstructing information (e.g., structure of loops). This type of information has mostly already been present in the code's source/intermediate representation prior to the lowering process to machine code. Like other authors [101, 140], this treatise argues that analysis techniques should be tightly integrated with programming languages and the subsequent compiler infrastructure. With this line of argumentation, the work on *vNV-Heap* [8] leverages programming-language features to support analysis-aware systems: Specifically, *vNV-Heap* makes use of the ownership memory-management model, which is, for example, inherent to the Rust programming language, to guarantee memory safety. Utilizing memory-safe languages like Rust will become increasingly important [143, 144] for building safe, secure, and resource-constrained embedded systems. *vNV-Heap*'s core idea is to track object modifications with the help of ownership and borrowing, being zero-cost, compile-time abstractions. By tracking these modifications during runtime and allowing upper bounds of the modified state in volatile memory, *vNV-Heap* enables the analysis to give worst-case resource-consumption bounds for persisting to non-volatile memory. Also, highly energy-constrained systems can rely on considerably large memory to be processed, for example, in the context of machine-learning algorithms with the necessity to update weights. In such memory-intensive scenarios, *vNV-Heap* is intended to support this increase in functionality with the possibility to manage memory under power-failure resilience.

Checkpointing Networking Stacks *vNV-Heap* is one example of a data structure and a runtime library that enables systems to persist the application's state when executing under possible power failures. Besides the application layer, the whole system stack also needs to be aware of the problem of power failures and, as a solution, store state to non-volatile memory. This requirement applies to all stateful networked embedded systems and includes the necessity to persistently store the state of the networking stack across blackout periods. In this context, *PfIP* introduces a transactional network stack for UDP/IP. For making *PfIP* an analyzable system, the stack relies on the representation of state machines, with annotated bounds and costs for each node and all corresponding transitions in between.

Towards Generic Fault Models Under Resource Constraints The motivation to increase functionality in stateful embedded system stacks also covers the systems' ability to handle arbitrary faults. The fault model of BFT not only covers system crashes but also allows revealing malicious systems when reaching consensus over a common state. Originally, BFT protocols were developed for server-grade platforms without the necessity to account for strict resource limits. As a consequence, directly using these protocols' implementations for resource-constrained embedded platforms has not been possible. To solve this problem, the work on *TinyBFT* [10] has been the first to demonstrate the use of a BFT protocol under severe resource bounds (i.e., less than $400\,\mathrm{kB}$ RAM). *TinyBFT* relies on the PBFT prototype [75] and introduced a protocol-aware memory-demand analysis. Based on the

analysis results, the configurability of data structures makes it possible to meet the given memory bounds. The work on *TinyBFT* underlines the problem of systems not designed with optimization- and analysis-awareness in mind, in terms of resource demands. Future protocols in this direction might not rely on retrofitted specialization mechanisms to meet resource limits.

From Analysis-Aware Systems to System-Wide Tailoring Having systems that are analysis-aware with respect to the constrained resource is the foundation for applying automated optimization techniques in order to tailor the system towards a specific objective. When systems are designed from scratch in a way that all required information on the time/energy/memory behavior is statically visible at the software-controlled interface, system-wide ahead-of-time optimizations are possible to provably minimize resource demands. Avoiding the need for retrofitted system abstractions (as is the case for *TinyBFT*), the following work on the *WatwaOS* analysis/optimization framework and generated OS is an example of such a ground-up approach.

5.2 Whole-System Specialization

Numerous embedded systems have both timing and energy limits. Examples include medical (implantable) devices that are battery-operated. Such systems further require developers to give runtime guarantees of task workloads while ensuring their efficiency.

Serving Diverse Workloads of Embedded Systems OSes play a central role in embedded systems since they have the purpose to serve the systems' workloads. The OS further contains the scheduler, which can be considered to represent the main component for resource management. Combined with the information of ahead-of-runtime analysis of tasks, as outlined in Chapter 3, the scheduler is eventually able to give runtime guarantees. For efficiency, the previously outlined clock subsystem in Section 4.2 is the hardware-related subsystem, controlled by the OS, for finding worst-case–optimal configurations. With the purpose of best serving application-level workloads, an understanding of their diversity is crucial. Section 3.3 emphasized the relevance of considering task workloads that heavily rely on the use of power-consuming devices, which are selectively activated. Such device-bound workload [63] may require the OS to start tasks only when sufficient energy is available in contexts of guaranteed execution under power-failure resilience. With the focus on efficiency, serving CPU-bound workloads with high-speed clock configuration and subsequently entering sleep states, also referred to as the *race-to-sleep strategy*, usually leads to increased energy efficiency. This improved efficiency stems from the fact that switching off components in sleep states reduces the static power demand, which usually has a more significant impact on the embedded SoCs' overall demand. In contrast to CPU-bound workloads, memory-bound jobs may have to rely on often comparably slow buses, which allows for throttling the speed. By employing this *crawl-to-sleep strategy*, energy can be saved for memory-bound workload phases.

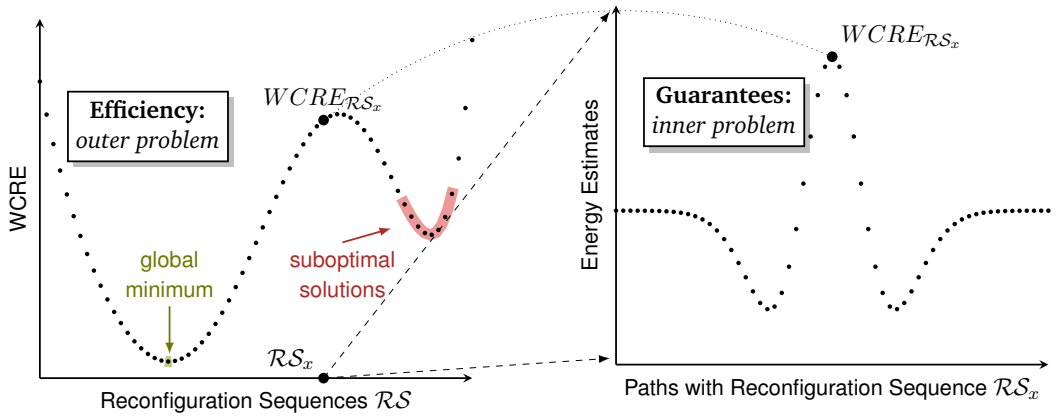

Figure 5.1: *WatwaOS*'s bilevel problem has an inner/maximizing problem (for guarantees) and an outer/minimizing problem (for guarantees). The objective is to determine worst-case–optimal configurations of the system. A specific reconfiguration sequence $\mathcal{RS}_x$ along a system-wide program path leads to a WCRE estimate $WCRE_{\mathcal{RS}_x}$.

Bilevel Optimization Problems for Worst-Case–Optimal Systems Under consideration of the diverse workloads (i.e., especially CPU- & memory-bound) in embedded systems, *WatwaOS* proposes an ahead-of-runtime approach for guaranteeing execution within resource budgets while provably ensuring efficiency. *WatwaOS*'s configurations on the clock subsystem are referred to as reconfiguration sequences, denoted with $\mathcal{RS}$. Under the consideration of complex, intertwined clock subsystems, the specialization of *WatwaOS* targets the minimization of the worst-case energy demand of tasks in the presence of synchronous and asynchronous interferences (i.e., interrupts and tasks of higher priority). Consequently, *WatwaOS* is concerned about finding reconfiguration sequences $\mathcal{RS}$ that provably optimize statically determined WCRE estimates. Equation (5.1) introduces the main optimization objective of *WatwaOS*, where $WCRE_{opt}$ refers to the final solution and $WCRE_{\mathcal{RS}}$ to a WCRE estimate under consideration of $\mathcal{RS}$:

$$WCRE_{opt} = \underbrace{\min_{\mathcal{RS}} \Big(\underbrace{\max \; (WCRE_{\mathcal{RS}})}_{\textit{inner problem}} \Big)}_{\textit{outer problem}} \tag{5.1}$$

Finding a solution for $WCRE_{opt}$ is referred to as a bilevel optimization problem, which is characterized by an inner and an outer problem. The inner problem concerns a maximization for having an upper bound on the WCRE, which addresses *WatwaOS*'s objective of providing runtime guarantees. For the outer problem, *WatwaOS* finds within the search space of possible reconfiguration sequences the one that minimizes the energy demand. Figure 5.1 gives an exemplary illustration of this problem. The eventual goal of the *WatwaOS* framework is to determine the actual global minimum (marked in green), instead of local minima (visualized in red). On the right side of Figure 5.1, the determination of the worst-case energy demand under consideration of a single $\mathcal{RS}_x$ is illustrated. On the

left side, the outer problem addresses the objective of finding the most energy-efficient configuration. Both sides use exemplary values for the inner maximization and the outer minimization problem. For determining these solutions, *WatwaOS* is able to use either multiple ILPs or one common bilevel formulation. Solving these problems, with the help of external optimization solvers (e.g., Gurobi [88] for ILPs and MIBS [158] for bilevel problems), eventually determines worst-case–optimal reconfiguration sequences.

Inserting Points of Reconfiguration From a theoretical point of view, a possible reconfiguration can be (automatically) inserted at arbitrary points of the program, for example, with the finest granularity of individual machine-code instructions. However, from a practical perspective, a too fine-grained introduction of reconfiguration points substantially increases the search space. *WatwaOS* allows for manually introducing reconfigurations and has possibilities to automatically introduce program points (e.g., on a task-based level, prior to loops or system calls). With an increasing number of possible points in $\mathcal{RS}$, the size of the ILP grows, in terms of variables and constraints, which, in turn, increases the ILP solving times.

Improving ILP Solving Fast ILP solving is crucial for numerous decision-making problems, as prevailing in *WatwaOS*. The framework employs techniques to keep the solving times reasonably low for practical usability on three levels: First, *WatwaOS* uses hierarchical abstractions that allow for utilizing similarities in the generated PSTGs. Second, to use similarities in the ILP formulation, *WatwaOS* exploits features of the utilized solver where single scenarios (i.e., reconfiguration sequences) can be evaluated with a common base ILP. Namely, this technique refers to Gurobi's support for multi-scenario ILPs [159]. Third, during the ILP solving process itself, *WatwaOS* has a mechanism based on callbacks of the ILP solver that allows the framework to terminate solving procedures prematurely when a suboptimal or infeasible interim result is detected. This termination can apply when a deadline violation or a higher energy demand than an already observed solution is found.

Generating Specialized Systems Software ecosystems surrounding an OS, which is with a specific focus on timeliness often referred to as a real-time operating system (RTOS), can be generally subdivided into dynamic and generative approaches. Dynamism gives flexibility to the implementation of the target application and the associated tooling infrastructure. Serving as an example for dynamic systems, FreeRTOS [160] gives developers the flexibility to dynamically instantiate thread objects. Besides this flexibility, a benefit for such types of dynamically instantiated systems is the fact that they have low requirements on the toolchain (e.g., choice of compiler) for building the whole system. However, this dynamic instantiation is in contrast to the structure of most embedded systems that usually have a fixed task set, which is known prior to runtime. Consequently, the dynamism can be an unneeded and unfavorable property of the embedded system. In this line of thought, Fiedler et al. [161] have shown that performance optimizations are possible during the system's (re)boot for dynamically created objects with a conceptually static nature. One step towards generative approaches first covers OSes that have a declarative description of application

requirements. ERIKA [162] serves as an example of a RTOS that is compliant with the OSEK standard [119]. These OSEK-compliant systems make use of a declarative configuration language, named OSEK Implementation Language (OIL). Combining the declarative description and the actual implementation of the application tasks' logic, with the help of specific tools (e.g., RT-Druid [163] for ERIKA), allows for generating the final system. Such types of generative systems conceptually still support the possibility of changing the compiler. The approach of *WatwaOS* goes one step further by supporting the generation of systems. Thereby, it sacrifices the support to change the compiler infrastructure for the goal of system specialization. The *WatwaOS* framework is directly integrated into the LLVM compiler infrastructure, as is the case for *WatwaOS*'s used *Platin* toolkit (see Section 3.5). For its whole-system analysis, *WatwaOS* makes use of the knowledge of the CFG of the application's tasks for deriving the PSTG abstraction. Eventually, *WatwaOS* generates an executable binary, which includes the specialized OS and the application tasks.

Main Results State-of-the-art operating systems lack abstractions that allow for ahead-of-runtime tailoring within the time-energy tradeoff under consideration of time and energy bounds. To address this gap, *WatwaOS* leverages PSTG abstractions and makes them aware of the ILP-solving process. That is, *WatwaOS* exploits similarities in the PSTG in order to reduce the size of the ILP problem and, as a consequence thereof, to reduce the time to find worst-case–optimal solutions. Further, exploiting several features specific to the ILP solving (e.g., multi-scenario ILPs) substantially reduced analysis times. Specifically, *WatwaOS* shows reductions of the ILP solving times of up to $90\,\%$. In terms of energy efficiency, the framework revealed substantial WCRE reductions (e.g., around $50\,\%$), while giving guarantees on the upper, end-to-end resource demand of executed tasks.

5.3 Towards Carbon-Aware Embedded Systems

Necessity of Carbon-Aware Budgeting The level of granularity at which technology handles resources in the domain of embedded systems can be considered to be impressive: In the temporal dimension, communication schemes, such as White Rabbit [164], achieve sub-nanosecond accuracy for synchronization. Optical communication works with pulses in the femtosecond time range [165]. In the dimension of energy, backscatter communication accounts for picojoules per bit. Unfortunately, considering resource budgeting and adherence to resource limits with a wider—literally global—perspective than on embedded systems shows a different picture: In view of CO_2 footprints, humanity is not (yet) able to adhere to given resource budgets. That is, under consideration of the earth's regenerative biocapacity, humanity has been overshooting the given budget for several decades [166, 167, 168]. While several metrics exist to quantify environmental impacts and the exceeding of planetary limits [169], this subsequent section and the related paper *CO_2CoDe* [12] on carbon-aware embedded systems make a simplifying assumption and describe environmental impact as CO_2-equivalent.

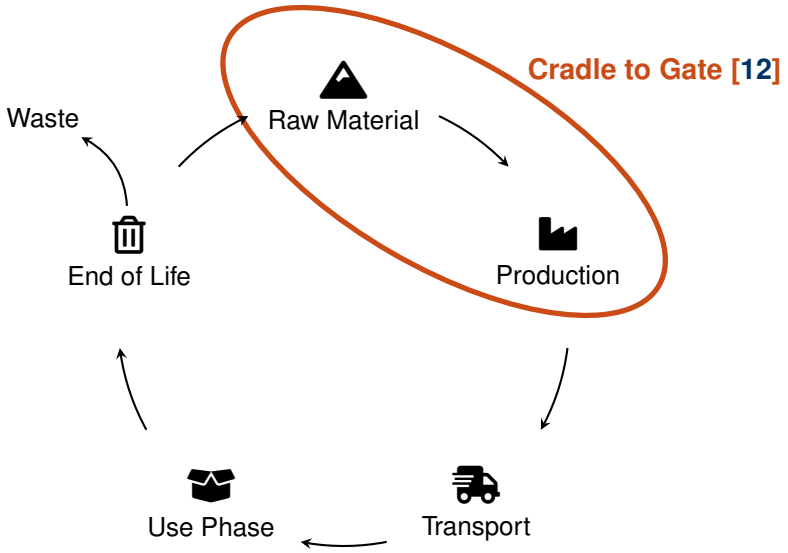

Figure 5.2: Handling the carbon footprint in an upfront assessment requires considerations of multiple phases of a system's life cycle [96].

Avoiding Shadow Cost: Problem of Embodied Carbon The Internet of Batteryless Things aims to drive sustainable systems by being free from batteries, since they usually use capacitors for storing the harvested energy. Indeed, intermittent systems harvest their operational energy from the environment and, as a consequence thereof, also have zero carbon footprint during their use phase ( ). As illustrated in Figure 5.2, this use phase is part of the generic life cycle of any system that has a carbon footprint. However, this perspective neglects the carbon footprint for producing these embedded systems, which is referred to as the embodied carbon footprint. In view of the life cycle, the embodied carbon footprint considered by *CO_2CoDe* accounts for the parts (,) highlighted in the orange ellipse. While intermittent systems shift their entire carbon footprint to the production phase, other mobile systems, such as smartphones, likewise demand the majority of their footprint prior to the use phase [78, 170, 171]. Given that the sector of information and communication technologies (ICT) contributes a significant amount to the global carbon footprint [172], the question arises whether turning knobs and tradeoffs exist for embedded systems. In this context, *CO_2CoDe* has a particular focus on intermittent systems.

Energy Storage & Cross-Layer Effects *CO_2CoDe* [12] looks at the energy storage of these systems and the implications of choosing different types with regard to their carbon footprint. Ruppel et al. [173] showed that the problem of parasitic effects in capacitors, which make them deviate from ideal components, propagates throughout the entire system stack. Thus, the choice of capacitors has an impact on the scheduling of tasks. Likewise, the capacitor choice also has an impact on the system's embodied carbon footprint. Consequently, for optimizing systems with both objectives of maximizing the software's energy efficiency and reducing the hardware's carbon footprint, carbon-aware co-design is necessary.

Carbon-Aware System-Level Co-Design The co-design approach of *CO_2CoDe* is in contrast to the previously outlined works, where specialization took place under a given hardware platform with fixed electronic components. Existing hardware/software co-design approaches in the domain of carbon-aware systems, such as ACT [174], employ hardware/-software co-design on the architectural level of processors. In contrast to ACT, *CO_2CoDe* presents the idea of carbon-aware co-design on the system level, with a focus on intermittent systems and their energy storage. That is, different types of capacitors with their electrical properties and carbon footprints are taken into account. When revisiting the static analysis approach for ensuring forward progress in device-driven intermittent systems from Section 3.3, the chosen set of capacitors requires the analysis to be aware of the overall circuit level along with its electronic components (as indicated with the symbol in the analysis/optimization workflow in Figure 3.2). Besides the paper on *CO_2CoDe* [12], Raffeck provides more insights into *CO_2CoDe* and the interplay with analysis techniques in his dissertation [63].

5.4 Main Systems-Related Papers

The following five papers comprise the main part of this chapter.

LCTES '25 Markus E. Gerber, Luis Gerhorst, Ishwar Mudraje, Kai Vogelgesang, Thorsten Herfet, and Peter Wägemann [8]
vNV-Heap: An Ownership-based Virtually Non-Volatile Heap for Embedded Systems
Proceedings of the 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2025)

CCNC '25 Kai Vogelgesang, Ishwar Mudraje, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, Thorsten Herfet, and Wolfgang Schröder-Preikschat [9]
PfIP: A UDP/IP Transactional Network Stack for Power-Failure Resilience in Embedded Systems
Proceedings of the 22nd IEEE Consumer Communications & Networking Conference (CCNC 2025)

RTAS '24 Harald Böhm, Tobias Distler, and Peter Wägemann [10]
TinyBFT: Byzantine Fault-Tolerant Replication for Highly Resource-Constrained Embedded Systems
Proceedings of the 30th Real-Time and Embedded Technology and Applications Symposium (RTAS '24)

RTSS '25 Tobias Häberlein, Eva Dengler, Phillip Raffeck, and Peter Wägemann [11]
WatwaOS: A Framework for Worst-Case–Aware Tailoring and Whole-System Analysis of Energy-Constrained Real-Time Systems
Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)

HotCarbon '24 Phillip Raffeck, Sven Posner, and Peter Wägemann [12]
CO2CoDe: Towards Carbon-Aware Hardware/Software Co-Design for Intermittently-Powered Embedded Systems
Proceedings of the 3rd HotCarbon Workshop on Sustainable Computer Systems (HotCarbon '24)

CHAPTER 6
OUTLOOK & CONCLUSION

Embedded systems are ubiquitous, and the challenge of having both efficient and safe execution under resource limits has been a relevant topic in the past and will be in the future, especially with regard to the increasing number of connected devices. This cumulative habilitation treatise provides a compilation of research works in the context of embedded systems within resource limits. This final chapter is structured as follows: First, the achieved results are summarized in Section 6.1. Subsequently, several directions of future work are outlined in Section 6.2. Finally, this treatise ends with concluding remarks (Section 6.3).

6.1 Summary of Results

Analysis-Related Results Having accurate analysis techniques and tooling support within the compilation infrastructure is a foundation of this habilitation treatise. Depending on the targeted application scenario, several analysis techniques are outlined that address different types of resources (i.e., time, energy) and different analysis types (i.e., static, dynamic). The contributions in this direction comprise, for example, the increase in accuracy with respect to device-driven [1] and temperature-dependent [4] embedded systems. Moreover, considering the real-time system's state for a fuzzing-based dynamic analysis in order to reduce analysis effort is a further result [5].

Optimization-Related Results DVFS approaches have been extensively explored in the context of real-time scheduling. However, these approaches have shortcomings with regard to the time-energy tradeoff that modern embedded SoCs offer. That is, the heterogeneity and complexity of clock subsystems offer a huge search space for worst-case–optimal (re-)configurations in this tradeoff. Main optimization-related results comprise finding such

solutions for strictly periodic real-time systems having solely synchronous task releases with the *FusionClock* [6] approach. However, most real-world systems come with the problem of serving asynchronous workloads due to the presence of asynchronous interrupts. To solve this problem, the work on *Crêpe* [7] provided an extension to the *FusionClock* approach and introduced software-controlled preemption control for systems with end-device requirements and intertwined clock subsystems.

Systems-Related Results In order to support the safe operation and to enhance the forward progress of whole systems with the requirement of power-failure resilience, worst-case analysis results can either be beneficial or inevitable. This type of resilience has to cover all aspects of a system's stack: In this line of thought, *vNV-Heap* [8] proposed a worst-case–aware runtime system for memory management and *PfIP* [9] a communication stack. Besides the requirement of power-failure resilience, *WatwaOS* [11], being one of the most recent works in this cumulative habilitation treatise, combines the three subject areas of (1) analysis, (2) optimization, and (3) systems with the largest overlap (see also illustration in Figure 1.2). The *WatwaOS* framework, which eventually generates a specialized operating system, achieves an analysis of the application's code and tailors it for the given hardware platform in terms of energy and time. Besides the approach to output a specialized OS, one avenue of future work is to integrate comparable abstractions into widely used OSes, as Dengler et al. have motivated for Zephyr [47]. However, retrofitting abstractions is a challenging endeavor in complex software stacks. For *WatwaOS*, both the analysis and the optimization techniques are designed to be system-aware and, vice versa, the system is aware of the analysis/optimization. This combined analysis/optimization/system consideration marks one of the main results of this cumulative habilitation treatise and its goal of achieving the highest subject-wise overlap, as illustrated with the goal arrows in Figure 1.2. Based on the previous analysis- and optimization-related results, the main system-related result of *WatwaOS* eventually combines the aspects of safety and runtime guarantees (i.e., inner problem of the bilevel formulation) with the aspect of resource-efficient execution (i.e., outer problem).

6.2 Future Work

Addressing Increasing Heterogeneity in Analyses The landscape of hardware used for embedded systems experiences an ongoing increase in heterogeneity, where the availability of intertwined clock subsystems provides (in Section 4.2) one example. Embedded SoCs, which can be used for low-power applications, nowadays have multiple symmetric processing cores, co-processors for low-power computing, or direct memory access (DMA) controllers with a comparably large feature set, allowing for chained computations [175, 176, 177]. The worst-case–aware tailoring under consideration of these heterogeneous computing units is an open topic of future research. When having data-intensive applications with the primary focus on performance rather than on low power and energy efficiency,

SoCs (e.g., Zynq UltraScale) exist that constitute very heterogeneous computing environments [178]: This is due to their extensive support of graphics processing units (GPUs) along with specialized co-processors/accelerators. Especially for numerous applications in the domain of machine learning, such GPU-centric platforms are relevant. For the WCET analysis of GPUs, research works exist that approach the problem of determining safe upper execution-time bounds [179, 180]. In line with the mentioned heterogeneity of compute units, future work can target the tailoring of combined CPU/GPU workloads with a focus on worst-case–optimality in the time-energy tradeoff. Comparable to the approach of *WatwaOS* (see Section 5.2) with its tailoring of workloads to the available hardware, serving tasks with possible GPU offloading is a challenging topic.

Hybrid Analysis Techniques The increasing heterogeneity and the accompanying increasing complexity of modern and COTS hardware platforms will make the problem of determining static resource-consumption models even more challenging. Presumably, the development of these models will not be able to keep up with the pace of the hardware's increasing complexity, which calls for alternative solutions. Building on previous work on dynamic, fuzzing-based worst-case response time analysis [5], one promising avenue is to combine results from such dynamic analyses with static analysis techniques: This combination is known as hybrid timing analysis. The commercial tool TimeWeaver [105] is an example of this type of analysis: It combines hardware performance traces with sound path-based reasoning with the IPET (see Section 2.1). Although the combination of unsound models and sound path analysis inevitably results in unsoundness, this hybrid approach is likely the only resort for future hardware platforms. This proposed direction of future work on hybrid whole-system timing analysis is to combine traces that were identified by fuzzing techniques like *FRET* (see Section 3.6.2) with system-wide ILP formulations, as used in the whole-system analysis and tailoring approach of *WatwaOS* [11].

Solving Strategies for Whole-System Specialization Especially the work on *WatwaOS* indicates that the step of ILP solving poses a bottleneck to the specialization approach. This bottleneck has also been previously identified in works on system-wide time and energy-consumption analyses [23, 26]. Although the *WatwaOS* approach already comprises several optimizations to reduce ILP-solving times, two further directions of future work exist, which potentially allow the analysis of more complex real-time systems: First, naïvely formulating ILPs without knowledge of the solver's strategy can result in poor performance (e.g., due to formulations using many unnecessary variables in the ILP). Having more awareness of the actual solving strategy and increasing the integration between the solver and the problem likely increases the performance. Second, loosening the requirement of finding an optimal solution offers possibilities to utilize different strategies: Specifically, using heuristic approaches, potentially reinforcement learning, could represent a suitable strategy for combined time, energy, and memory limits.

6.3 Concluding Remarks

Towards Carbon-Aware Embedded Systems This cumulative habilitation treatise presented a compilation of several contributions in the domain of embedded systems with the three overlapping topics of analysis technique, optimization techniques, and their integration into the systems themselves. The majority of the presented topics cover time, energy, and memory constraints. Only a single work [12] specifically targets the challenge of carbon-aware systems. The ICT sector is moving at a fast pace, and the advances, for example, of AI technologies come at a high price in terms of energy and carbon footprint [181, 182]. This treatise presented several works that address resource budgeting at a fine granularity (i.e., $\mu\,\mathrm{sec}$, $\mu\mathrm{J}$, bit). When zooming out from this fine-grained perspective to a bigger picture of resource demands, the fact that humanity has been overshooting resource budgets and planetary limits, unfortunately, becomes apparent [166, 167, 168, 169]. From my personal perspective, the overarching challenge of climate change outweighs efficiency-related topics in terms of social relevance and must be tackled in an interdisciplinary way. I am convinced that life-cycle thinking and treating carbon as a first-class resource are essential directions for a sustainable future of embedded systems within global resource limits.

LIST OF ACRONYMS

ABB	atomic basic block
ADC	analog-to-digital converter
BB	basic block
BFT	Byzantine fault tolerance
CFG	control-flow graph
COTS	commercial-off-the-shelf
DAC	digital-to-analog converter
DNN	deep neural network
DMA	direct memory access
DVFS	dynamic voltage and frequency scaling
GPU	graphics processing unit
GWP	global warming potential
IC	integrated circuit
ICT	information and communication technologies
ILP	integer linear program
IoT	Internet of Things
IoBT	Internet of Batteryless Things
IoMT	Internet of Medical Things
IPET	implicit path-enumeration technique
LCA	life cycle assessment
LLVM IR	LLVM intermediate representation
LoRa	long range

NVM	non-volatile memory
OIL	OSEK Implementation Language
OS	operating system
PABB	power atomic basic block
PSTG	power-state–transition graph
PML	Program Metainfo Language
PLL	phase-locked loop
QP	quadratic program
RTOS	real-time operating system
SoC	system-on-chip
TFLM	TensorFlow Lite Micro
STG	state-transition graph
WCEC	worst-case energy consumption
WCET	worst-case execution time
WCRE	worst-case response energy consumption
WCRT	worst-case response time
WOEC	worst-observed energy consumption
WOET	worst-observed execution time
WORE	worst-observed response energy consumption
WORT	worst-observed response time

APPENDIX A
BIBLIOGRAPHY

A.1 General Bibliography

[1] Phillip Raffeck, Johannes Maier, and Peter Wägemann. „WoCA: Avoiding Intermittent Execution in Embedded Systems by Worst-Case Analyses with Device States.“ In: *Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '24)*. 2024, pages 83–94. DOI: 10.1145/3652032.3657569. URL: https://sys.cs.fau.de/publications/2024/raffeck_24_lctes.pdf.

[2] Ishwar Mudraje, Jasper Devreker, Kai Vogelgesang, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, and Thorsten Herfet. „Reverse Engineering the ESP32-C3 Wi-Fi Drivers for Static Worst-Case Analysis of Intermittently-Powered Systems.“ In: *Proceedings of the 13th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems (ENSsys '25)*. 2025. DOI: 10.1145/3722572.3727926. URL: https://dl.acm.org/doi/10.1145/3722572.3727926.

[3] Emad Jacob Maroun, Eva Dengler, Christian Dietrich, Stefan Hepp, Henriette Herzog, Benedikt Huber, Jens Knoop, Daniel Wiltsche-Prokesch, Peter Puschner, Phillip Raffeck, Martin Schoeberl, Simon Schuster, and Peter Wägemann. „The Platin Multi-Target Worst-Case Analysis Tool.“ In: *Proceedings of the 22nd International Workshop on Worst-Case Execution Time Analysis (WCET '24)*. 2024. URL: https://drops.dagstuhl.de/storage/01oasics/oasics-vol121-wcet2024/OASIcs.WCET.2024.2/OASIcs.WCET.2024.2.pdf.

[4] Kilian Müller, Johannes Weidner, Norman Franchi, and Peter Wägemann. „TinyEP: TinyML-enhanced Energy Profiling for Extreme Edge Devices.“ In: *IEEE Access* 12 (2024), pages 193747–193762. DOI: 10.1109/ACCESS.2024.3520089. URL: https://doi.org/10.1109/ACCESS.2024.3520089.

[5] Alwin Berger, Simon Schuster, Peter Wägemann, and Peter Ulbrich. „Dynamic Fuzzing-Based Whole-System Timing Analysis.“ In: *Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)*. 2025. URL: https://sys-sideshow.cs.tu-dortmund.de/publications/berger_25_rtss.pdf.

[6] Eva Dengler, Phillip Raffeck, Simon Schuster, and Peter Wägemann. „FusionClock: Energy-Optimal Clock-Tree Reconfigurations for Energy-Constrained Real-Time Systems.“ In: *Proceedings of the 35th Euromicro Conference on Real-Time Systems (ECRTS '23)*. Volume 262. 2023, 6:1–6:24. URL: https://drops.dagstuhl.de/opus/volltexte/2023/18035/pdf/LIPIcs-ECRTS-2023-6.pdf.

[7] Eva Dengler and Peter Wägemann. „Crêpe: Clock-Reconfiguration–Aware Preemption Control in Real-Time Systems with Devices.“ In: *Proceedings of the 36th Euromicro Conference on Real-Time Systems (ECRTS '24)*. ***Best Paper Award***. 2024. URL: https://drops.dagstuhl.de/storage/00lipics/lipics-vol298-ecrts2024/LIPIcs.ECRTS.2024.10/LIPIcs.ECRTS.2024.10.pdf.

[8] Markus E. Gerber, Luis Gerhorst, Ishwar Mudraje, Kai Vogelgesang, Thorsten Herfet, and Peter Wägemann. „vNV-Heap: An Ownership-based Virtually Non-Volatile Heap for Embedded Systems.“ In: *Proceedings of the 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2025)*. 2025. URL: https://arxiv.org/pdf/2501.17707.

[9] Kai Vogelgesang, Ishwar Mudraje, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, Thorsten Herfet, and Wolfgang Schröder-Preikschat. „PfIP: A UDP/IP Transactional Network Stack for Power-Failure Resilience in Embedded Systems.“ In: *Proceedings of the 22nd IEEE Consumer Communications & Networking Conference (CCNC 2025)*. Las Vegas, USA, Jan. 2025, page 6. DOI: 10.1109/CCNC54725.2025.10975988. URL: https://sys.cs.fau.de/publications/2025/vogelgesang_25_ccnc.pdf.

[10] Harald Böhm, Tobias Distler, and Peter Wägemann. „TinyBFT: Byzantine Fault-Tolerant Replication for Highly Resource-Constrained Embedded Systems.“ In: *Proceedings of the 30th Real-Time and Embedded Technology and Applications Symposium (RTAS '24)*. 2024. DOI: 10.1109/RTAS61025.2024.00026. URL: https://sys.cs.fau.de/publications/2024/boehm_24_rtas.pdf.

[11] Tobias Häberlein, Eva Dengler, Phillip Raffeck, and Peter Wägemann. „WatwaOS: A Framework for Worst-Case–Aware Tailoring and Whole-System Analysis of Energy-Constrained Real-Time Systems.“ In: *Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)*. 2025. URL: https://sys.cs.fau.de/publications/2025/haeberlein_25_rtss.pdf.

[12] Phillip Raffeck, Sven Posner, and Peter Wägemann. „CO2CoDe: Towards Carbon-Aware Hardware/Software Co-Design for Intermittently-Powered Embedded Systems.“ In: *Proceedings of the 3rd HotCarbon Workshop on Sustainable Computer Systems (HotCarbon '24)*. 2024. DOI: 10.1145/3727200.3727219. URL: https://sys.cs.fau.de/publications/2024/raffeck_24_hotcarbon.pdf.

[13] Markus E. Gerber, Luis Gerhorst, Ishwar Mudraje, Kai Vogelgesang, Thorsten Herfet, and Peter Wägemann. „Artifact: vNV-Heap: An Ownership-based Virtually Non-Volatile Heap for Embedded Systems.“ In: *Peer-Reviewed Artifact Evaluation, 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2025)*. 2025. DOI: 10.1145/3712433.

[14] Eva Dengler and Peter Wägemann. „Crêpe: Clock Reconfigurability for Preemption Control (Artifact).“ In: *Dagstuhl Artifacts Series* 10.1 (2024), 2:1–2:3. ISSN: 2509-8195. DOI: 10.4230/DARTS.10.1.2. URL: https://drops.dagstuhl.de/entities/document/10.4230/DARTS.10.1.2.

[15] Eva Dengler, Phillip Raffeck, Simon Schuster, and Peter Wägemann. „FusionClock: WCEC-Optimal Clock-Tree Reconfigurations (Artifact).“ In: *Dagstuhl Artifacts Series* 9.1 (2023), 2:1–2:3. ISSN: 2509-8195. DOI: 10.4230/DARTS.9.1.2. URL: https://drops.dagstuhl.de/entities/document/10.4230/DARTS.9.1.2.

[16] Kilian Müller, Siddharth Mane, Peter Wägemann, and Norman Franchi. „WasmWeaver: A Framework for Runtime-Aware WebAssembly Program Generation with Reinforcement Learning.“ In: *Proceedings of the 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering, Tool Demo Track (SANER Tool Demo 2026)*. 2026. URL: https://sys.cs.fau.de/publications/2026/mueller_26_saner-tool.pdf.

[17] Maximilian Seidler, Martin Michelis, Peter Wägemann, and Rüdiger Kapitza. „Wasm-WCET: Worst-Case Execution-Time Analysis of WebAssembly Modules on Updatable Resource-Constrained Embedded Devices.“ In: *Proceedings of the 31st Real-Time and Embedded Technology and Applications Symposium (RTAS '26)*. 2026. URL: https://sys.cs.fau.de/publications/2026/seidler_26_rtas.pdf.

[18] Luis Gerhorst, Henriette Herzog, Peter Wägemann, Maximilian Ott, Rüdiger Kapitza, and Timo Hönig. „VeriFence: Lightweight and Precise Spectre Defenses for Untrusted Linux Kernel Extensions.“ In: *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '24)*. RAID '24. Association for Computing Machinery, 2024, pages 644–659. DOI: 10.1145/3678890.3678907. URL: https://arxiv.org/pdf/2405.00078.

[19] Bernhard Heinloth, Peter Wägemann, and Wolfgang Schröder-Preikschat. „LUCI – Loader-based Dynamic Software Updates for Off-the-shelf Shared Objects.“ In: *Proceedings of the 2023 USENIX Annual Technical Conference (USENIX ATC '23)*. 2023, pages 1–15. URL: https://sys.cs.fau.de/publications/2023/heinloth_23_atc.pdf.

[20] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Annotate Once – Analyze Anywhere: Context-Aware WCET Analysis by User-Defined Abstractions.“ In: *Proceedings of the 22nd ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '21)*. 2021, pages 54–66. DOI: 10.1145/3461648.3463847. URL: https://www4.cs.fau.de/Publications/2021/schuster_21_lctes.pdf.

[21] Steffen Vaas, Peter Ulbrich, Christian Eichler, Peter Wägemann, Marc Reichenbach, and Dietmar Fey. „Taming Non-Deterministic Low-Level I/O: Predictable Multi-Core Real-Time Systems by SoC Co-Design.“ In: *Proceedings of the 24th IEEE International Symposium on Real-Time Distributed Computing (ISORC '21)*. 2021, pages 43–52. DOI: 10.1109/ISORC52013.2021.00017. URL: https://www4.cs.fau.de/Publications/2021/vaas_21_isorc.pdf.

[22] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Proving Real-Time Capability of Generic Operating Systems by System-Aware Timing Analysis.“ In: *Proceedings of the 25th Real-Time and Embedded Technology and Applications Symposium (RTAS '19)*. 2019, pages 318–330. DOI: 10.1109/RTAS.2019.00034. URL: https://www4.cs.fau.de/Publications/2019/schuster_19_rtas.pdf.

[23] Peter Wägemann, Christian Dietrich, Tobias Distler, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Whole-System Worst-Case Energy-Consumption Analysis for Energy-Constrained Real-Time Systems.“ In: *Proceedings of the 30th Euromicro Conference on Real-Time Systems (ECRTS '18)*. Volume 106. ***Outstanding Paper Award***. 2018, 24:1–24:25. DOI: 10.4230/LIPIcs.ECRTS.2018.24. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8979.

[24] Peter Wägemann, Tobias Distler, Christian Eichler, and Wolfgang Schröder-Preikschat. „Benchmark Generation for Timing Analysis." In: *Proceedings of the 23rd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*. 2017, pages 319–330. DOI: 10.1109/RTAS.2017.6. URL: https://www4.cs.fau.de/Publications/2017/waegemann_17_rtas.pdf.

[25] V. Sieh, R. Burlacu, T. Hönig, H. Janker, P. Raffeck, Peter Wägemann, and W. Schröder-Preikschat. „An End-to-End Toolchain: From Automated Cost Modeling to Static WCET and WCEC Analysis." In: *Proceedings of the 20th International Symposium on Real-Time Distributed Computing (ISORC '17)*. ***Best Paper Award***. 2017, pages 1–10. DOI: 10.1109/ISORC.2017.10. URL: https://www4.cs.fau.de/Publications/2017/sieh_17_isorc.pdf.

[26] Christian Dietrich, Peter Wägemann, Peter Ulbrich, and Daniel Lohmann. „SysWCET: Whole-System Response-Time Analysis for Fixed-Priority Real-Time Systems." In: *Proceedings of the 23rd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*. ***Outstanding Paper Award***. 2017, pages 37–48. DOI: 10.1109/RTAS.2017.37. URL: https://www4.cs.fau.de/Publications/2017/dietrich_17_rtas.pdf.

[27] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, and Volkmar Sieh. „A Kernel for Energy-Neutral Real-Time Systems with Mixed Criticalities." In: *Proceedings of the 22nd Real-Time and Embedded Technology and Applications Symposium (RTAS '16)*. 2016, pages 25–36. DOI: 10.1109/RTAS.2016.7461320. URL: https://www4.cs.fau.de/Publications/2016/waegemann_16_rtas.pdf.

[28] Peter Wägemann, Tobias Distler, Timo Hönig, Heiko Janker, Rüdiger Kapitza, and Wolfgang Schröder-Preikschat. „Worst-Case Energy Consumption Analysis for Energy-Constrained Embedded Systems." In: *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS '15)*. 2015, pages 105–114. DOI: 10.1109/ECRTS.2015.17. URL: https://www4.cs.fau.de/Publications/2015/waegemann_15_ecrts.pdf.

[29] Peter Wägemann. „Energieverbrauchsanalyse mittels impliziter Pfadaufzählung und genetischer Algorithmen." In: *Betriebssysteme und Echtzeit*. Proceedings of the Conference Echtzeit '15. 2015, pages 109–118. DOI: 10.1007/978-3-662-48611-5_12.

[30] Timo Hönig, Christopher Eibel, Benedict Herzog, Heiko Janker, Peter Wägemann, and Wolfgang Schröder-Preikschat. „Playing Hare and Tortoise: The FigarOS Kernel for Fine-Grained System-Level Energy Optimizations." In: *Proceedings of the 5th Brazilian Symposium on Computing Systems Engineering (SBESC '15)*. 2015, pages 80–83. URL: https://www4.cs.fau.de/Publications/2015/hoenig_15_sbesc.pdf.

[31] Simon Ripperger, Gerald Carter, Rachel Page, Niklas Duda, Alexander Kölpin, Robert Weigel, Markus Hartmann, Thorsten Nowak, Jörn Thielecke, Michael Schadhauser, Jörg Robert, Sebastian Herbst, Klaus Meyer-Wegener, Peter Wägemann, Wolfgang Schröder-Preikschat, Björn Cassens, Rüdiger Kapitza, Falko Dressler, and Frieder Mayer. „Thinking small: next-generation sensor networks close the size gap in vertebrate biologging." In: *PLOS Biology* 18.4 (2020), pages 1–25. DOI: 10.1371/journal.pbio.3000655. URL: https://doi.org/10.1371/journal.pbio.3000655.

[32] V. Sieh, R. Burlacu, T. Hönig, H. Janker, P. Raffeck, Peter Wägemann, and W. Schröder-Preikschat. „Combining Automated Measurement-Based Cost Modeling with Static Worst-Case Execution-Time and Energy-Consumption Analyses." In: *IEEE Embedded Systems Letters* 11.2 (2019), pages 38–41. DOI: 10.1109/LES.2018.2868823. URL: https://www4.cs.fau.de/Publications/2019/sieh_19_esl.pdf.

[33] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, Volkmar Sieh, and Wolfgang Schröder-Preikschat. „Operating Energy-Neutral Real-Time Systems.“ In: *ACM Transactions on Embedded Computing Systems (ACM TECS)* 17.1 (2018), 11:1–11:25. DOI: 10.1145/3078631. URL: https://www4.cs.fau.de/Publications/2017/waegemann_17_tecs.pdf.

[34] Niklas Duda, Thorsten Nowak, Markus Hartmann, Michael Schadhauser, Björn Cassens, Peter Wägemann, Muhammad Nabeel, Simon Ripperger, Sebastian Herbst, Klaus Meyer-Wegener, Frieder Mayer, Falko Dressler, Wolfgang Schröder-Preikschat, Rüdiger Kapitza, Jörg Robert, Jörn Thielecke, Robert Weigel, and Alexander Koelpin. „BATS: Adaptive Ultra Low Power Sensor Network for Animal Tracking.“ In: *Sensors* 18.10 (2018), pages 1–34. DOI: 10.3390/s18103343. URL: https://doi.org/10.3390/s18103343.

[35] Sebastian Maier, Timo Hönig, Peter Wägemann, and Wolfgang Schröder-Preikschat. „Asynchronous Abstract Machines: Anti-noise System Software for Many-core Processors.“ In: *Proceedings of the 9th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS '19)*. 2019, pages 16–26. DOI: 10.1145/3322789.3328744. URL: https://www4.cs.fau.de/Publications/2019/maier_19_ross.pdf.

[36] Phillip Raffeck, Christian Eichler, Peter Wägemann, and Wolfgang Schröder-Preikschat. „Worst-Case Energy-Consumption Analysis by Microarchitecture-Aware Timing Analysis for Device-Driven Cyber-Physical Systems.“ In: *Proceedings of the 19th International Workshop on Worst-Case Execution Time Analysis (WCET '19)*. 2019, 6:1–6:12. DOI: 10.4230/OASIcs.WCET.2019.4. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10769.

[37] Christian Eichler, Peter Wägemann, and Wolfgang Schröder-Preikschat. „GenEE: A Benchmark Generator for Static Analysis Tools of Energy-Constrained Cyber-Physical Systems.“ In: *Proceedings of the 2nd Workshop on Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench '19)*. 2019. DOI: 10.1145/3312480.3313170. URL: https://www4.cs.fau.de/Publications/2019/eichler_19_cpsiotbench.pdf.

[38] Christian Eichler, Tobias Distler, Peter Ulbrich, Peter Wägemann, and Wolfgang Schröder-Preikschat. „TASKers: A Whole-System Generator for Benchmarking Real-Time-System Analyses.“ In: *Proceedings of the 18th International Workshop on Worst-Case Execution Time Analysis (WCET '18)*. 2018, 6:1–6:12. ISBN: 978-3-95977-073-6. DOI: 10.4230/OASIcs.WCET.2018.6. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9752.

[39] Heiko Falk, Sebastian Altmeyer, Peter Hellinckx, Björn Lisper, Wolfgang Puffitsch, Christine Rochange, Martin Schoeberl, Rasmus Bo Sørensen, Peter Wägemann, and Simon Wegener. „TACLeBench: A Benchmark Collection to Support Worst-Case Execution Time Research.“ In: *Proceedings of the 16th International Workshop on Worst-Case Execution Time Analysis (WCET '16)*. 2016, pages 1–10. ISBN: 978-3-95977-025-5. DOI: 10.4230/OASIcs.WCET.2016.2. URL: http://drops.dagstuhl.de/opus/volltexte/2016/6895.

[40] Peter Wägemann, Tobias Distler, Timo Hönig, Volkmar Sieh, and Wolfgang Schröder-Preikschat. „GenE: A Benchmark Generator for WCET Analysis.“ In: *Proceedings of the 15th International Workshop on Worst-Case Execution Time Analysis (WCET '15)*. Volume 47. Dagstuhl, Germany, 2015, pages 33–43. ISBN: 978-3-939897-95-8. DOI: 10.4230/OASIcs.WCET.2015.33. URL: http://drops.dagstuhl.de/opus/volltexte/2015/5254.

[41] Kai Vogelgesang, Phillip Raffeck, Peter Wägemann, Thorsten Herfet, and Wolfgang Schröder-Preikschat. „WIP: Towards a Transactional Network Stack for Power-Failure Resilience.“ In: *Proceedings of the 21st IEEE Consumer Communications & Networking Conference (CCNC WiP '24) - Work-In-Progress*. 2024. DOI: 10.1109/CCNC51664.2024.10454781. URL: https://sys.cs.fau.de/publications/2024/vogelgesang_24_ccnc_wip.pdf.

[42] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Towards System-Wide Timing Analysis of Real-Time–Capable Operating Systems.“ In: *Proceedings of the 30th Euromicro Conference on Real-Time Systems Work-in-Progress Session (ECRTS '18 WiP)*. 2018, pages 1–3. URL: https://www4.cs.fau.de/Publications/2018/schuster_18_ecrts-wip.pdf.

[43] Peter Wägemann, Tobias Distler, Phillip Raffeck, and Wolfgang Schröder-Preikschat. „Poster Abstract: Towards Code Metrics for Benchmarking Timing Analysis.“ In: *Proceedings of the 37th Real-Time Systems Symposium (RTSS '16)*. 2016, page 369. DOI: 10.1109/RTSS.2016.048. URL: https://www4.cs.fau.de/Publications/2016/waegemann_16_rtss.pdf.

[44] Peter Wägemann, Tobias Distler, Phillip Raffeck, and Wolfgang Schröder-Preikschat. „Towards Code Metrics for Benchmarking Timing Analysis.“ In: *Proceedings of the 37th Real-Time Systems Symposium Work-in-Progress Session (RTSS WiP '16)*. 2016. URL: https://www4.cs.fau.de/Publications/2016/waegemann_16_rtss-wip.pdf.

[45] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. *PragMetis Source Code*. Source code of the related LCTES publication titled: Annotate Once - Analyze Anywhere: Context-Aware WCET Analysis by User-Defined Abstractions. Apr. 2021. DOI: 10.5281/zenodo.4697392. URL: https://doi.org/10.5281/zenodo.4697392.

[46] Peter Wägemann, Christian Dietrich, Tobias Distler, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Whole-System WCEC Analysis for Energy-Constrained Real-Time Systems (Artifact).“ In: *Dagstuhl Artifacts Series* 4.2 (2018), 7:1–7:4. DOI: 10.4230/DARTS.4.2.7. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8975.

[47] Eva Dengler, Tobias Häberlein, Phillip Raffeck, and Peter Wägemann. „Positional Paper: A Clock-Distribution Abstraction for Resource-Constrained Real-Time Systems in Zephyr.“ In: *Proceedings of the 1st International Conference on Zephyr in Science and Education (ZiSE '25)*. 2025. URL: https://sys.cs.fau.de/publications/2025/2025-dengler-zephyr-sceduconf.pdf.

[48] Bernhard Heinloth, Peter Wägemann, and Wolfgang Schröder-Preikschat. „LUCI – Loader-based Dynamic Software Updates for Off-the-shelf Shared Objects.“ In: *Poster Session of the 2023 USENIX Annual Technical Conference (USENIX ATC '23)*. Poster. 2023, page 1. URL: https://sys.cs.fau.de/publications/2023/heinloth_23_atc-poster.pdf.

[49] Phillip Raffeck, Johannes Maier, and Peter Wägemann. *WoCA Source Code: Hard- & Software Prototype*. 2025. DOI: 10.5281/zenodo.14811325. URL: https://zenodo.org/records/14811326.

[50] Phillip Raffeck and Peter Wägemann. *Ecology-Aware Material Use as a Pervasive Trait in Intermittent Real-Time Systems*. Poster Session/Real-Time Pitches Session of the 35th Euromicro Conference on Real-Time Systems (ECRTS '23). Abstract: https://sys.cs.fau.de/publications/2023/raffeck_23_ecrts-abstract.pdf & Poster: https://sys.cs.fau.de/publications/2023/raffeck_23_ecrts-poster.pdf. Vienna, Austria, 2023.

[51] Alwin Berger, Simon Schuster, Peter Wägemann, and Peter Ulbrich. *OS-State–Aware Fuzzing for Worst-Case Response Times*. Abstract: https://www.betriebssysteme.org/wp-content/uploads/2022/09/FGBS2022-autumn_paper_343.pdf, Talk: https://www.betriebssysteme.org/wp-content/uploads/2022/09/FGBS2022-autumn_slides_343.pdf. Fachgruppentreffen Erlangen, Fachgruppe Betriebssysteme der Gesellschaft für Informatik (GI SYS, FG BS), September 2022. 2022.

[52] Peter Wägemann and Simon Ripperger. *Software: Ground node design for BATS tracking system*. https://doi.naturkundemuseum.berlin/data/10.7479/z5ym-kx58. 2020. DOI: 10.7479/z5ym-kx58.

[53] Peter Wägemann, Florian Harbecke, Bernhard Heinloth, Henriette Hofmeier, and Wolfgang Schröder-Preikschat. *An Energy-Neutral, WiFi-Connected Room Display with Hand-Crank–Based Energy Harvesting*. Implementation Award, 2nd place, FAU Ideenwettbewerb (FAU idea competition), proposal: https://www4.cs.fau.de/Publications/2019/waegemann_19_fau_ideenwettbewerb.pdf. 2019.

[54] Christian Dietrich and Peter Wägemann. „SysWCET: Ende-zu-Ende-Antwortzeiten für OSEK-Systeme." In: *Proceedings of the Embedded Software Engineering Kongress (ESE '17)*. 2017. URL: https://sra.uni-hannover.de/Publications/2017/dietrich_17_ese.pdf.

[55] Christian Eichler, Peter Wägemann, Tobias Distler, and Wolfgang Schröder-Preikschat. „Demo Abstract: Tooling Support for Benchmarking Timing Analysis." In: *Proceedings of the 23rd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*. 2017, pages 159–160. DOI: 10.1109/RTAS.2017.20. URL: https://www4.cs.fau.de/Publications/2017/eichler_17_rtas-demo.pdf.

[56] Christopher Eibel, Sebastian Herbst, Björn Cassens, Timo Hönig, Peter Wägemann, Heiko Janker, Rüdiger Kapitza, Klaus Meyer-Wegener, and Wolfgang Schröder-Preikschat. *A Flexible, Adaptive System for Data-Stream Processing in Energy-Constrained Ad-hoc Networks*. Technical report. Erlangen: Friedrich-Alexander University Erlangen-Nürnberg (FAU), 2016, pages 1–8. URL: https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/6893.

[57] Peter Wägemann, Timo Hönig, Rüdiger Kapitza, and Wolfgang Schröder-Preikschat. *Worst-Case Energy Consumption Analysis for Soft and Hard Energy Systems*. Poster Session at the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14). 2014.

[58] Peter Wägemann. „Energy-Constrained Real-Time Systems and Their Worst-Case Analyses." PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2020. URL: https://nbn-resolving.org/urn:nbn:de:bvb:29-opus4-146935.

[59] Peter Wägemann. „Static Worst-Case Analyses and Their Validation Techniques for Safety-Critical Systems." In: *Ernst Denert Award for Software Engineering 2020*. 2022, pages 227–247. URL: https://doi.org/10.1007/978-3-030-83128-8_11.

[60] Peter Wägemann. „Energiebeschränkte Echtzeitsysteme und ihre Worst-Case-Analysen.“ In: *Ausgezeichnete Informatikdissertationen 2020*. Gesellschaft für Informatik e.V., 2021, pages 329–338. URL: http://dl.gi.de/handle/20.500.12116/37920.

[61] Peter Wägemann, editor. *21st International Workshop on Worst-Case Execution Time Analysis (WCET 2023)*. Volume 114. OASIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. ISBN: 978-3-95977-293-8. URL: https://drops.dagstuhl.de/entities/volume/OASIcs-volume-114.

[62] Doğanalp Ergenç, Shadi Attarha, and Peter Wägemann, editors. *Proceedings of the 1st Workshop on Resilient Networks and Systems (ReNeSys 2025)*. Berlin, Germany, 2025. DOI: 10.14279/depositonce-24399.

[63] Phillip Raffeck. „Predictable Execution for Device-Bound Embedded Systems with Intermittent Power.“ PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2025. DOI: 10.25593/open-fau-2372. URL: https://doi.org/10.25593/open-fau-2372.

[64] Brian Santo. „Embedded Battle Royal.“ In: *IEEE Spectrum* 38.12 (2001), pages 36–41.

[65] Peter Marwedel. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*. Fourth Edition. Springer, 2021. DOI: 10.1007/978-3-030-60910-8.

[66] Mark Weiser. „The Computer for the 21st Century.“ In: *Scientific American* 265.3 (1991), pages 94–105. URL: https://www.jstor.org/stable/pdf/24938718.pdf.

[67] IoT Analytics GmbH. *State of IoT 2025*. 2025. URL: https://iot-analytics.com/number-connected-iot-devices/.

[68] Saad Ahmed, Bashima Islam, Kasim Sinan Yildirim, Marco Zimmerling, Przemysław Pawełczak, Muhammad Hamad Alizai, Brandon Lucia, Luca Mottola, Jacob Sorber, and Josiah Hester. „The Internet of Batteryless Things.“ In: *Communications of the ACM* 67.3 (Feb. 2024), pages 64–73. ISSN: 0001-0782. DOI: 10.1145/3624718. URL: https://doi.org/10.1145/3624718.

[69] Jasper Devreker. *Unveiling secrets of the ESP32: creating an open-source MAC Layer*. 2023. URL: https://zeus.ugent.be/blog/23-24/open-source-esp32-wifi-mac/.

[70] Jasper Devreker. *ESP32 open MAC*. 2025. URL: https://esp32-open-mac.be/.

[71] Pete Warden and Daniel Situnayake. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019.

[72] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, and Pete Warden. „TensorFlow Lite Micro: Embedded machine learning for TinyML systems.“ In: *Proceedings of Machine Learning and Systems 3 (MLSys 2021)*. Volume 3. 2021, pages 800–811.

[73] Johannes Weidner. „Hybrid Integrated Real-Time Energy Measurement with TinyML.“ Master's thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Technische Fakultät, 2024.

[74] Christian Dietrich, Martin Hoffmann, and Daniel Lohmann. In: *ACM Transactions on Embedded Computing Systems (ACM TECS* (Jan. 2, 2017).

[75] M. Castro and B. Liskov. „Practical Byzantine Fault Tolerance.“ In: *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI '99)*. 1999, pages 173–186.

[76] Michel Rottleuthner, Thomas C. Schmidt, and Matthias Wahlisch. „Dynamic Clock Reconfiguration for the Constrained IoT and Its Application to Energy-Efficient Networking.“ In: *Proceedings of the International Conference On Embedded Wireless Systems And Networks (EWSN '22)*. 2023, pages 168–179. DOI: 10.48550/arXiv.2102.10353.

[77] Holly Chiang, Hudson Ayers, Daniel Giffin, Amit Levy, and Philip Levis. „Power Clocks: Dynamic Multi-Clock Management for Embedded Systems.“ In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN '21)*. 2021, pages 139–150. DOI: 10.1145/3274783.3275176.

[78] Pol Maistriaux, Thibault Pirson, Maxime Schramme, Jérôme Louveaux, and David Bol. „Modeling the Carbon Footprint of Battery-Powered IoT Sensor Nodes for Environmental-Monitoring Applications.“ In: *Proceedings of the 12th International Conference on the Internet of Things (IoT '22)*. 2022, pages 9–16. DOI: 10.1145/3567445.3567448.

[79] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. „The FAIR Guiding Principles for scientific data management and stewardship.“ In: *Scientific Data* 3.1 (2016), pages 1–9.

[80] Patrick Cousot. „La vérification des programmes par interprétation abstraite.“ Séminaire de la Chaire d'Innovation technologique – Liliane Bettencourt, Collège de France. 2008.

[81] Patrick Cousot and Radhia Cousot. „Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.“ In: *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. POPL '77. 1977, pages 238–252. DOI: 10.1145/512950.512973.

[82] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. „The Worst-case Execution-time Problem – Overview of Methods and Survey of Tools.“ In: *ACM Transactions on Embedded Computing Systems (ACM TECS)* 7.3 (2008), pages 1–53.

[83] Ramkumar Jayaseelan, Tulika Mitra, and Xianfeng Li. „Estimating the Worst-Case Energy Consumption of Embedded Software.“ In: *Proceedings of the 12th Real-Time and Embedded Technology and Applications Symposium (RTAS '06)*. 2006, pages 81–90. DOI: 10.1109/RTAS.2006.17.

[84] Peter Puschner. „Zeitanalyse von Echtzeitprogrammen.“ PhD thesis. Technische Universität Wien, 1993.

[85] P. Puschner and A. Schedl. „Computing Maximum Task Execution Times: A Graph-Based Approach.“ In: *Real-Time Systems* 13 (1997), pages 67–91.

[86] Yau-Tsun Steven Li and Sharad Malik. „Performance Analysis of Embedded Software Using Implicit Path Enumeration.“ In: *ACM SIGPLAN Notices*. Volume 30. 11. ACM. 1995, pages 88–98.

[87] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. *lp_solve 5.5, open source (mixed-integer) linear programming system. Software*. 2004.

[88] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2025. URL: https://docs.gurobi.com/projects/optimizer/en/current/index.html.

[89] IBM. *ILOG CPLEX Optimization Studio*. 2025. URL: https://www.ibm.com/products/ilog-cplex-optimization-studio.

[90] Infineon Technologies AG. *XMC4500 Reference Manual*. 2012.

[91] Freescale Semiconductor, Inc. *Kinetis KL46 Sub-Family*. 2014, pages 1–66.

[92] Semtech Corporation. *SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver*. Revision 7. 2020.

[93] Semtech Corporation. *Errata Note – SX1276/77/78 – 137 to 1020 MHz Low Power Long Range Transceiver*. Revision 1. 2013.

[94] STMicroelectronics. *Arm Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces, and camera*. Revision 10. 2024.

[95] Hari Cherupalli, Henry Duwe, Weidong Ye, Rakesh Kumar, and John Sartori. „Determining Application-Specific Peak Power and Energy Requirements for Ultra-Low-Power Processors." In: *ACM Transactions on Computer Systems (ACM TOCS)* 35.3 (2017), 9:1–9:33. DOI: 10.1145/3148052.

[96] Thibault Pirson. „Environmental perspectives and limits for the Internet of Things in the Anthropocene: going beyond systematic Life-Cycle Assessment." PhD thesis. UCL-Université Catholique de Louvain, 2023.

[97] GrammaTech, Inc. „NASA Study Explores the Benefits of Static Analysis." In: (). URL: https://codesecure.com/wp-content/uploads/2023/03/GrammaTech-NASA_White_Sands-Case_Study-CSO-19.pdf.

[98] Christian Ferdinand and Reinhold Heckmann. „aiT: Worst-Case Execution Time Prediction by Static Program Analysis." In: *Building the Information Society* 156 (2004), pages 377–383.

[99] AbsInt Angewandte Informatik GmbH. *aiT Worst-Case Execution Time Analyzers*. 2025. URL: https://web.archive.org/web/20250924134148/https://www.absint.com/ait/index.htm.

[100] C. Ballabriga, H. Cassé, C. Rochange, and P. Sainrat. „OTAWA: An Open Toolbox for Adaptive WCET Analysis." In: *Proceedings of the 8th International Workshop on Software Technolgies for Embedded and Ubiquitous Systems (SEUS '10)*. 2010, pages 35–46.

[101] H. Falk and P. Lokuciejewski. „A Compiler Framework for the Reduction of Worst-case Execution Times." In: *Real-time Systems* 46.2 (2010), pages 251–300. DOI: 10.1007/s11241-010-9101-x.

[102] Sebastian Hahn, Michael Jacobs, Nils Hölscher, Kuan-Hsun Chen, Jian-Jia Chen, and Jan Reineke. „LLVMTA: An LLVM-Based WCET Analysis Tool." In: *Proceedings of the 20th International Workshop on Worst-Case Execution Time Analysis (WCET '22)*. Edited by Clément Ballabriga. 2022, 2:1–2:17.

[103] D. Hardy, B. Rouxel, and I. Puaut. „The Heptane Static Worst-Case Execution Time Estimation Tool." In: *Proceedings of the 17th International Workshop on Worst-Case Execution Time Analysis (WCET '17)*. 2017, 8:1–8:12. DOI: 10.4230/OASIcs.WCET.2017.8. URL: http://drops.dagstuhl.de/opus/volltexte/2017/7303.

[104] N. Holsti and S. Saarinen. „Status of the Bound-T WCET Tool." In: *Proceedings of the 2nd International Workshop on Worst-Case Execution Time Analysis (WCET '02)*. 2002, pages 36–41.

[105] Daniel Kästner, Markus Pister, Simon Wegener, and Christian Ferdinand. „TimeWeaver: A Tool for Hybrid Worst-Case Execution Time Analysis." In: *Proceedings of the 19th International Workshop on Worst-Case Execution Time Analysis (WCET '19)*. 2019, 1:1–1:11. DOI: 10.4230/OASIcs.WCET.2019.1. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10766.

[106] Raimund Kirner. „The WCET Analysis Tool CalcWcet167.“ In: *Proceedings of the International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA '12)*. 2012, pages 158–172.

[107] Xianfeng Li, Yun Liang, Tulika Mitra, and Abhik Roychoudhury. „Chronos: A Timing Analyzer for Embedded Software.“ In: *Science of Computer Programming* 69.1 (2007), pages 56–67.

[108] B. Lisper. „SWEET – A Tool for WCET Flow Analysis.“ In: *Proceedings of the 6th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA '14)*. 2014, pages 482–485.

[109] James Pallister, Steve Kerrison, Jeremy Morse, and Kerstin Eder. „Data Dependent Energy Modeling for Worst Case Energy Consumption Analysis.“ In: *Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems (SCOPES '17)*. 2017, pages 51–59.

[110] Simon Wegener, Kris K. Nikov, Jose Nunez-Yanez, and Kerstin Eder. „EnergyAnalyzer: Using Static WCET Analysis Techniques to Estimate the Energy Consumption of Embedded Applications.“ In: *Proceedings of the 21th International Workshop on Worst-Case Execution Time Analysis (WCET '23)*. 2023, 9:1–9:14. DOI: 10.4230/OASIcs.WCET.2023.9. URL: https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.WCET.2023.9.

[111] Fabian Scheler. „Atomic Basic Blocks: Eine Abstraktion für die gezielte Manipulation der Echtzeitsystemarchitektur.“ PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Technische Fakultät, 2011. URL: http://opus4.kobv.de/opus4-fau/files/1740/FabianSchelerDissertation.pdf.

[112] Fabian Scheler, Martin Mitzlaff, and Wolfgang Schröder-Preikschat. „Atomic Basic Blocks.“ In: *PEARL Workshop 2007: Mobilität und Echtzeit*. 2007. ISBN: 978-3-540-74836-6.

[113] Fabian Scheler, Daniel Lohmann, Olaf Spinczyk, and Wolfgang Schröder-Preikschat. „Aspect-Oriented Real-Time Architecture—AORTA.“ In: *Proceedings of the 27th IEEE International Symposium on Real-Time Systems (RTSS '06)*. Work-in-Progress Session. 2006, pages 5–8.

[114] Fabian Scheler and Wolfgang Schröder-Preikschat. „Synthesising Real-Time Systems from Atomic Basic Blocks.“ In: *12th IEEE International Symposium on Real-Time and Embedded Technology and Applications (RTAS'06)*. Work-in-Progress Session. 2006, pages 49–52.

[115] Fabian Scheler and Wolfgang Schröder-Preikschat. „The Real-Time Systems Compiler: Migrating event-triggered systems to time-triggered systems.“ In: *Software: Practice and Experience* 41.12 (2011), pages 1491–1515. DOI: 10.1002/spe.1099.

[116] V. Wolfe, S. Davidson, and I. Lee. „RTC: language support for real-time concurrency.“ In: *Proceedings of the 12th Real-Time Systems Symposium (RTSS 1991)*. 1991, pages 43–52. DOI: 10.1109/REAL.1991.160357.

[117] Karsten Albers, Frank Bodmann, and Frank Slomka. „Hierarchical event streams and event dependency graphs: A new computational model for embedded real-time systems.“ In: *Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS 2006)*. 2006, pages 1–10. DOI: 10.1109/ECRTS.2006.12.

[118] Emilia Farcas, Claudiu Farcas, Wolfgang Pree, and Josef Templ. „Transparent distribution of real-time components based on logical execution time.“ In: *Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2005)*. 2005, pages 31–39. DOI: 10.1145/1065910.1065915.

[119] OSEK/VDX Group. *Operating System Specification 2.2.3*. Technical report. OSEK/VDX Group, 2005.

[120] Christian Dietrich, Martin Hoffmann, and Daniel Lohmann. „Cross-Kernel Control-Flow-Graph Analysis for Event-Driven Real-Time Systems.“ In: *Proceedings of the Conference on Languages, Compilers and Tools for Embedded Systems (LCTES '15)*. 2015, 6:1–6:10.

[121] Christian Dietrich, Martin Hoffmann, and Daniel Lohmann. „Global Optimization of Fixed-Priority Real-Time Systems by RTOS-Aware Control-Flow Analysis.“ In: *ACM Transactions on Embedded Computing Systems (ACM TECS)* 16.16 (2 2017), 35:1–35:25. DOI: 10.1145/2950053.

[122] M. Hoffmann, F. Lukas, C. Dietrich, and D. Lohmann. „dOSEK: The Design and Implementation of a Dependability-Oriented Static Embedded Kernel.“ In: *Proceedings of the 21st Real-Time and Embedded Technology and Applications Symposium (RTAS '15)*. 2015, pages 259–270. URL: https://www4.cs.fau.de/Publications/2015/hoffmann_15_rtas.pdf.

[123] Tobias Klaus, Matthias Becker, Wolfgang Schröder-Preikschat, and Peter Ulbrich. „Constrained Data-Age with Job-Level Dependencies: How to Reconcile Tight Bounds and Overheads.“ In: *Proceedings of the 27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2021)*. 2021, pages 66–79. DOI: 10.1109/RTAS52030.2021.00014.

[124] Gerion Entrup, Andreas Kässens, Björn Fiedler, and Daniel Lohmann. „Applied static analysis and specialization of cross-core syscalls for multi-core AUTOSAR OS.“ In: *Real-Time Systems* 60.3 (2024), pages 491–533. DOI: 10.1007/s11241-024-09429-1.

[125] Bahram Yarahmadi and Erven Rohou. „Compiler optimizations for safe insertion of checkpoints in intermittently powered systems.“ In: *Proceedings of the 20th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS '20)*. 2020, pages 169–185. DOI: 10.1007/978-3-030-60939-9_12.

[126] Jongouk Choi, Larry Kittinger, Qingrui Liu, and Changhee Jung. „Compiler-Directed High-Performance Intermittent Computation with Power Failure Immunity.“ In: *28th Real-Time and Embedded Technology and Applications Symposium (RTAS '22)*. 2022, pages 40–54. DOI: 10.1109/RTAS54340.2022.00012.

[127] Espressif Systems Co., Ltd. *ESP32-C3 Technical Reference Manual*. Version 1.3. 2022. URL: https://www.espressif.com/sites/default/files/documentation/esp32-c3_technical_reference_manual_en.pdf.

[128] Vivek Tiwari, Sharad Malik, Andrew Wolfe, and Mike Tien-Chien Lee. „Instruction Level Power Analysis and Optimization of Software.“ In: *Journal of VLSI Signal Processing Systems* 13.2-3 (1996), pages 223–238.

[129] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel. „An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations.“ In: *Proceedings of the International Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS '01)*. 2001.

[130] Jeremy Morse, Steve Kerrison, and Kerstin Eder. „On the Limitations of Analyzing Worst-Case Dynamic Energy of Processing.“ In: *ACM Transactions on Embedded Computing Systems (ACM TECS)* 17.3 (2018), 59:1–59:22. DOI: 10.1145/3173042. URL: http://doi.acm.org/10.1145/3173042.

[131] Texas Instruments. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*.

[132] Simon Kellner and Frank Bellosa. „Energy Accounting Support in TinyOS.“ In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 32.2 (2009), pages 105–109.

[133] Hai-Ying Zhou, Dan-Yan Luo, Yan Gao, and De-Cheng Zuo. „Modeling of Node Energy Consumption for Wireless Sensor Networks.“ In: *Wireless Sensor Network* 3.1 (2011), page 18.

[134] Domenico Balsamo, Alex S Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M Al-Hashimi, Geoff V Merrett, and Luca Benini. „Hibernus++: a self-calibrating and adaptive system for transiently-powered embedded devices.“ In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.12 (2016), pages 1968–1980. DOI: 10.1109/TCAD.2016.2547919.

[135] Gautier Berthou, Pierre-Évariste Dagand, Delphine Demange, Rémi Oudin, and Tanguy Risset. „Intermittent computing with peripherals, formally verified.“ In: *Proceedings of the 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '20)*. 2020, pages 85–96.

[136] Hrishikesh Jayakumar, Arnab Raha, and Vijay Raghunathan. „QuickRecall: A low overhead HW/SW approach for enabling computations across power cycles in transiently powered computers.“ In: *Proceedings of the 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems (VLSID '14)*. 2014, pages 330–335. DOI: 10.1109/VLSID.2014.63.

[137] Kiwan Maeng and Brandon Lucia. „Supporting Peripherals in Intermittent Systems with Just-in-time Checkpoints.“ In: *Proceedings of the 40th Conference on Programming Language Design and Implementation (PLDI '19)*. 2019, pages 1101–1116. DOI: 10.1145/3314221.3314613. URL: http://doi.acm.org/10.1145/3314221.3314613.

[138] B. Huber, D. Prokesch, and P. Puschner. „Combined WCET Analysis of Bitcode and Machine Code Using Control-flow Relation Graphs.“ In: *Proceedings of the 14th Conference on Languages, Compilers and Tools for Embedded Systems (LCTES '13)*. 2013, pages 163–172.

[139] Stefan Hepp, Benedikt Huber, Jens Knoop, Daniel Prokesch, and Peter P. Puschner. „The platin Tool Kit - The T-CREST Approach for Compiler and WCET Integration.“ In: *Proceedings 18th Kolloquium Programmiersprachen und Grundlagen der Programmierung, KPS 2015, Pörtschach, Austria, October 5-7, 2015*. 2015. URL: http://www.complang.tuwien.ac.at/kps2015/proceedings/KPS_2015_submission_49.pdf.

[140] Peter Puschner, Daniel Prokesch, Benedikt Huber, Jens Knoop, Stefan Hepp, and Gernot Gebhard. „The T-CREST Approach of Compiler and WCET-Analysis Integration.“ In: *Proceedings of the 16th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2013)*. 2013, pages 1–8. DOI: 10.1109/ISORC.2013.6913220.

[141] Chris Lattner and Vikram Adve. „LLVM: A compilation framework for lifelong program analysis & transformation.“ In: *Proceedings of the International Symposium on Code Generation and Optimization (CGO '04)*. 2004, pages 75–86.

[142] Stefan Pfahler. „Towards Energy-Aware Byzantine Fault-Tolerant DAG-Based Consensus.“ Master's thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Technische Fakultät, 2025.

[143] Alex Rebert and Christoph Kern. *Secure by Design: Google's Perspective on Memory Safety*. Technical report. Google Security Engineering, 2024. URL: https://research.google/pubs/secure-by-design-googles-perspective-on-memory-safety/.

[144] *Back to the Building Blocks: A Path Toward Secure and Measurable Software*. Technical report. The Office of the White House, 2024. URL: https://bidenwhitehouse.archives.gov/wp-content/uploads/2024/02/Final-ONCD-Technical-Report.pdf.

[145] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, S. Hepp, B. Huber, A. Jordan, E. Kasapaki, J. Knoop, Y. Li, D. Prokesch, W. Puffitsch, P. Puschner, A. Rocha, C. Silva, J. Sparsø, and A. Tocchi. „T-CREST: Time-predictable Multi-Core Architecture for Embedded Systems." In: *Journal of Systems Architecture* 61.9 (2015), pages 449–471.

[146] M. Schoeberl, F. Brandner, S. Hepp, W. Puffitsch, and D. Prokesch. *Patmos Reference Handbook*. Technical report. Technical University of Denmark, 2014.

[147] Renato Mancuso, Rodolfo Pellizzoni, Marco Caccamo, Lui Sha, and Heechul Yun. „WCET(m) Estimation in Multi-Core Systems using Single Core Equivalence." In: *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRT '15)*. 2015, pages 174–183. DOI: 10.1109/ECRTS.2015.23.

[148] Bjorn Andersson, Dionisio de Niz, Gabriel Moreno, Jeffery Hansen, Mark Klein, et al. *Assessing the Use of Machine Learning to Find the Worst-Case Execution Time of Avionics Software*. Technical report. United States. Department of Transportation. Federal Aviation Administration (FAA). William J. Hughes Technical Center, 2023. DOI: https://doi.org/10.21949/1528215.

[149] D. Trilla, C. Hernandez, J. Abella, and F. J. Cazorla. „Worst-Case Energy Consumption: A New Challenge for Battery-Powered Critical Devices." In: *IEEE Transactions on Sustainable Computing* (2019), pages 1–8.

[150] Barton P. Miller, Louis Fredriksen, and Bryan So. „An empirical study of the reliability of UNIX utilities." In: *Communications of the ACM* 33.12 (Dec. 1, 1990), pages 32–44. ISSN: 0001-0782. DOI: 10.1145/96267.96279. URL: https://doi.org/10.1145/96267.96279 (visited on 05/24/2022).

[151] Xiaogang Zhu, Sheng Wen, Seyit Camtepe, and Yang Xiang. „Fuzzing: A Survey for Roadmap." In: *ACM Comput. Surv.* 54.11s (2022). ISSN: 0360-0300. DOI: 10.1145/3512345. URL: https://doi.org/10.1145/3512345.

[152] Michal Zalewski. *american fuzzy lop*. 2025. URL: https://lcamtuf.coredump.cx/afl/.

[153] Andrea Fioraldi, Dominik Maier, Dongjia Zhang, and Davide Balzarotti. „LibAFL: A Framework to Build Modular and Reusable Fuzzers." In: *Proceedings of the 29th ACM Conference on Computer and Communications Security (CCS '22)*. ACM, 2022.

[154] STMicroelectronics. *Ultra-low-power with FPU Arm Cortex-M4 MCU 80 MHz with 1 Mbyte of Flash memory, LCD, USB OTG, DFSDM*. 2025. URL: https://www.st.com/resource/en/datasheet/stm32l476rg.pdf.

[155] Mario Bambagini, Mauro Marinoni, Hakan Aydin, and Giorgio Buttazzo. „Energy-Aware Scheduling for Real-Time Systems: A Survey." In: *ACM Transactions on Embedded Computing Systems (ACM TECS)* 15.1 (2016), 7:1–7:34. DOI: 10.1145/2808231.

[156] Daniel Kästner and Stephan Wilhelm. „Generic Control Flow Reconstruction from Assembly Code." In: volume 37. 2002, pages 46–55. DOI: 10.1145/566225.513839.

[157] H. Theiling. „Extracting Safe and Precise Control Flow from Binaries." In: *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications (RTCSA '00)*. 2000, pages 23–30. DOI: 10.1109/RTCSA.2000.896367.

[158] Sahar Tahernejad, Ted K Ralphs, and Scott T DeNegre. „A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation.“ In: *Mathematical Programming Computation* 12.4 (2020), pages 529–568.

[159] Gurobi Optimization, LLC. *Multiple Scenarios*. 2024. URL: https://web.archive.org/web/20241209001548/https://docs.gurobi.com/projects/optimizer/en/current/features/multiscenario.html.

[160] *FreeRTOS – Real-time operating system for microcontrollers and small microprocessors*. 2025. URL: https://freertos.org/.

[161] Björn Fiedler, Gerion Entrup, Christian Dietrich, and Daniel Lohmann. „ARA: Static Initialization of Dynamically-Created System Objects.“ In: *Proceedings of the 27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '21)*. 2021, pages 400–412. DOI: 10.1109/RTAS52030.2021.00039.

[162] Evidence Srl., ReTiS Lab, others. *Erika Enterprise | Open Source RTOS OSEK/VDX Kernel*. 2025. URL: https://erika.tuxfamily.org.

[163] Paolo Gai, Giuseppe Lipari, Marco Di Natale, Nicola Serreli, Luigi Palopoli, and Alberto Ferrari. *Adding timing analysis to functional design to predict implementation errors*. Technical report. SAE Technical Paper, 2007. URL: https://doi.org/10.4271/2007-01-1272.

[164] Maciej Lipiński, Tomasz Włostowski, Javier Serrano, and Pablo Alvarez. „White rabbit: A PTP application for robust sub-nanosecond synchronization.“ In: *Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS '11)*. 2011, pages 25–30. DOI: 10.1109/ISPCS.2011.6070148.

[165] Andrew D Ludlow, Martin M Boyd, Jun Ye, Ekkehard Peik, and Piet O Schmidt. „Optical Atomic Clocks.“ In: *Reviews of Modern Physics* 87.2 (2015), pages 637–701.

[166] David Lin, Leopold Wambersie, and Mathis Wackernagel. *Estimating the Date of Earth Overshoot Day 2023*. Technical report. Global Footprint Network, 2023. URL: https://www.overshootday.org/content/uploads/2023/06/Earth-Overshoot-Day-2023-Nowcast-Report.pdf.

[167] Mathis Wackernagel, David Lin, Mikel Evans, Laurel Hanscom, and Peter Raven. „Defying the Footprint Oracle: Implications of Country Resource Trends.“ In: *Sustainability* 11.7 (Apr. 2019). DOI: 10.3390/su11072164.

[168] Linus Blomqvist, Barry W. Brook, Erle C. Ellis, Peter M. Kareiva, Ted Nordhaus, and Michael Shellenberger. „Does the Shoe Fit? Real versus Imagined Ecological Footprints.“ In: *PLOS Biology* 11.11 (Nov. 2013), pages 1–6. DOI: 10.1371/journal.pbio.1001700. URL: https://doi.org/10.1371/journal.pbio.1001700.

[169] Katherine Richardson, Will Steffen, Wolfgang Lucht, Jørgen Bendtsen, Sarah E. Cornell, Jonathan F. Donges, Markus Drüke, Ingo Fetzer, Govindasamy Bala, Werner von Bloh, Georg Feulner, Stephanie Fiedler, Dieter Gerten, Tom Gleeson, Matthias Hofmann, Willem N. Huiskamp, Matti Kummu, Chinchu Mohan, David Nogués-Bravo, Stefan Petri, Miina Porkka, Stefan Rahmstorf, Sibyll Schaphoff, Kirsten Thonicke, Arne Tobian, Vili Virkki, Lan Wang-Erlandsson, Lisa Weber, and Johan Rockström. „Earth beyond six of nine planetary boundaries.“ In: *Science Advances* 9.37 (2023), pages 1–16. DOI: 10.1126/sciadv.adh2458.

[170] Mine Ercan, Jens Malmodin, Pernilla Bergmark, Emma Kimfalk, and Ellinor Nilsson. „Life cycle assessment of a smartphone.“ In: *ICT for Sustainability 2016*. 2016, pages 124–133. DOI: 10.2991/ict4s-16.2016.15.

[171] Thibault Pirson and David Bol. „Assessing the embodied carbon footprint of IoT edge devices with a bottom-up life-cycle approach.“ In: *Journal of Cleaner Production* 322 (Nov. 2021). ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2021.128966. URL: https://www.sciencedirect.com/science/article/pii/S0959652621031577.

[172] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S Blair, and Adrian Friday. „The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations.“ In: *Patterns* 2.9 (Aug. 2021). DOI: 10.1016/j.patter.2021.100340.

[173] Emily Ruppel, Milijana Surbatovich, Harsh Desai, Kiwan Maeng, and Brandon Lucia. „An Architectural Charge Management Interface for Energy-Harvesting Systems.“ In: *Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO '22)*. 2022, pages 318–335. DOI: 10.1109/MICRO56248.2022.00034.

[174] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. „ACT: designing sustainable computer systems with an architectural carbon modeling tool.“ In: *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA '22)*. 2022, pages 784–799. DOI: 10.1145/3470496.3527408.

[175] Espressif Systems Co., Ltd. *ESP32 Technical Reference Manual*. Version 5.6. 2025. URL: https://documentation.espressif.com/esp32_technical_reference_manual_en.pdf.

[176] Michael Rushanan and Stephen Checkoway. „Run-dma.“ In: *Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT '15)*. 2015. URL: https://www.usenix.org/system/files/conference/woot15/woot15-paper-rushanan.pdf.

[177] Thomas Benz, Michael Rogenmoser, Paul Scheffler, Samuel Riedel, Alessandro Ottaviano, Andreas Kurth, Torsten Hoefler, and Luca Benini. „A high-performance, energy-efficient modular DMA engine architecture.“ In: *IEEE Transactions on Computers* 73.1 (2023), pages 263–277. DOI: 10.1109/TC.2023.3329930.

[178] AMD. *Zynq UltraScale+ Device Technical Reference Manual*. Version 2.5. 2025. URL: https://docs.amd.com/v/u/en-US/ug1085-zynq-ultrascale-trm.

[179] Louison Jeanmougin, Thomas Carle, and Christine Rochange. „Bounding the WCET of a GPU Thread Block with a Multi-Phase Representation of Warps Execution.“ In: *Proceedings of the 37th Euromicro Conference on Real-Time Systems (ECRTS 2025)*. 2025, 11:1–11:26. DOI: 10.4230/LIPIcs.ECRTS.2025.11.

[180] Noïc Crouzet, Thomas Carle, and Christine Rochange. „Time-predictable warp scheduling in a GPU.“ In: *Microprocessors and Microsystems* (2025), pages 1–13.

[181] James O'Donnell and Casey Crownhart. *We did the math on AI's energy footprint. Here's the story you haven't heard.* 2025. URL: https://web.archive.org/web/20251022202226/https://www.technologyreview.com/2025/05/20/1116327/ai-energy-usage-climate-footprint-big-tech/.

[182] Jens Gröger, Felix Behrens, Peter Gailhofer, and Inga Hilbert. *Environmental Impacts of Artificial Intelligence*. en. Technical report. 2025. URL: https://www.oeko.de/fileadmin/oekodoc/Report_KI_ENG.pdf.

A.2 Personal Bibliography

All papers (i.e., conference, journal, workshop publications) that are listed subsequently were formally published, unless marked otherwise. The formal publishing means that these papers passed a scientific peer-reviewing process. After acceptance for publication, these papers appeared in the respective proceedings and/or were added to established digital libraries (i.e., ACM, IEEE, Dagstuhl).

Conferences

[1] Phillip Raffeck, Johannes Maier, and Peter Wägemann. „WoCA: Avoiding Intermittent Execution in Embedded Systems by Worst-Case Analyses with Device States.“ In: *Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '24)*. 2024, pages 83–94. DOI: 10.1145/3652032.3657569. URL: https://sys.cs.fau.de/publications/2024/raffeck_24_lctes.pdf.

[5] Alwin Berger, Simon Schuster, Peter Wägemann, and Peter Ulbrich. „Dynamic Fuzzing-Based Whole-System Timing Analysis.“ In: *Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)*. 2025. URL: https://sys-sideshow.cs.tu-dortmund.de/publications/berger_25_rtss.pdf.

[6] Eva Dengler, Phillip Raffeck, Simon Schuster, and Peter Wägemann. „FusionClock: Energy-Optimal Clock-Tree Reconfigurations for Energy-Constrained Real-Time Systems.“ In: *Proceedings of the 35th Euromicro Conference on Real-Time Systems (ECRTS '23)*. Volume 262. 2023, 6:1–6:24. URL: https://drops.dagstuhl.de/opus/volltexte/2023/18035/pdf/LIPIcs-ECRTS-2023-6.pdf.

[7] Eva Dengler and Peter Wägemann. „Crêpe: Clock-Reconfiguration–Aware Preemption Control in Real-Time Systems with Devices.“ In: *Proceedings of the 36th Euromicro Conference on Real-Time Systems (ECRTS '24)*. ***Best Paper Award***. 2024. URL: https://drops.dagstuhl.de/storage/00lipics/lipics-vol298-ecrts2024/LIPIcs.ECRTS.2024.10/LIPIcs.ECRTS.2024.10.pdf.

[8] Markus E. Gerber, Luis Gerhorst, Ishwar Mudraje, Kai Vogelgesang, Thorsten Herfet, and Peter Wägemann. „vNV-Heap: An Ownership-based Virtually Non-Volatile Heap for Embedded Systems.“ In: *Proceedings of the 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2025)*. 2025. URL: https://arxiv.org/pdf/2501.17707.

[9] Kai Vogelgesang, Ishwar Mudraje, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, Thorsten Herfet, and Wolfgang Schröder-Preikschat. „PfIP: A UDP/IP Transactional Network Stack for Power-Failure Resilience in Embedded Systems.“ In: *Proceedings of the 22nd IEEE Consumer Communications & Networking Conference (CCNC 2025)*. Las Vegas, USA, Jan. 2025, page 6. DOI: 10.1109/CCNC54725.2025.10975988. URL: https://sys.cs.fau.de/publications/2025/vogelgesang_25_ccnc.pdf.

[10] Harald Böhm, Tobias Distler, and Peter Wägemann. „TinyBFT: Byzantine Fault-Tolerant Replication for Highly Resource-Constrained Embedded Systems.“ In: *Proceedings of the 30th Real-Time and Embedded Technology and Applications Symposium (RTAS '24)*. 2024. DOI: 10.1109/RTAS61025.2024.00026. URL: https://sys.cs.fau.de/publications/2024/boehm_24_rtas.pdf.

[11] Tobias Häberlein, Eva Dengler, Phillip Raffeck, and Peter Wägemann. „WatwaOS: A Framework for Worst-Case–Aware Tailoring and Whole-System Analysis of Energy-Constrained Real-Time Systems.“ In: *Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)*. 2025. URL: https://sys.cs.fau.de/publications/2025/haeberlein_25_rtss.pdf.

[16] Kilian Müller, Siddharth Mane, Peter Wägemann, and Norman Franchi. „WasmWeaver: A Framework for Runtime-Aware WebAssembly Program Generation with Reinforcement Learning.“ In: *Proceedings of the 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering, Tool Demo Track (SANER Tool Demo 2026)*. 2026. URL: https://sys.cs.fau.de/publications/2026/mueller_26_saner-tool.pdf.

[17] Maximilian Seidler, Martin Michelis, Peter Wägemann, and Rüdiger Kapitza. „Wasm-WCET: Worst-Case Execution-Time Analysis of WebAssembly Modules on Updatable Resource-Constrained Embedded Devices.“ In: *Proceedings of the 31st Real-Time and Embedded Technology and Applications Symposium (RTAS '26)*. 2026. URL: https://sys.cs.fau.de/publications/2026/seidler_26_rtas.pdf.

[18] Luis Gerhorst, Henriette Herzog, Peter Wägemann, Maximilian Ott, Rüdiger Kapitza, and Timo Hönig. „VeriFence: Lightweight and Precise Spectre Defenses for Untrusted Linux Kernel Extensions.“ In: *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '24)*. RAID '24. Association for Computing Machinery, 2024, pages 644–659. DOI: 10.1145/3678890.3678907. URL: https://arxiv.org/pdf/2405.00078.

[19] Bernhard Heinloth, Peter Wägemann, and Wolfgang Schröder-Preikschat. „LUCI – Loader-based Dynamic Software Updates for Off-the-shelf Shared Objects.“ In: *Proceedings of the 2023 USENIX Annual Technical Conference (USENIX ATC '23)*. 2023, pages 1–15. URL: https://sys.cs.fau.de/publications/2023/heinloth_23_atc.pdf.

[20] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Annotate Once – Analyze Anywhere: Context-Aware WCET Analysis by User-Defined Abstractions.“ In: *Proceedings of the 22nd ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '21)*. 2021, pages 54–66. DOI: 10.1145/3461648.3463847. URL: https://www4.cs.fau.de/Publications/2021/schuster_21_lctes.pdf.

[21] Steffen Vaas, Peter Ulbrich, Christian Eichler, Peter Wägemann, Marc Reichenbach, and Dietmar Fey. „Taming Non-Deterministic Low-Level I/O: Predictable Multi-Core Real-Time Systems by SoC Co-Design.“ In: *Proceedings of the 24th IEEE International Symposium on Real-Time Distributed Computing (ISORC '21)*. 2021, pages 43–52. DOI: 10.1109/ISORC52013.2021.00017. URL: https://www4.cs.fau.de/Publications/2021/vaas_21_isorc.pdf.

[22] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Proving Real-Time Capability of Generic Operating Systems by System-Aware Timing Analysis.“ In: *Proceedings of the 25th Real-Time and Embedded Technology and Applications Symposium (RTAS '19)*. 2019, pages 318–330. DOI: 10.1109/RTAS.2019.00034. URL: https://www4.cs.fau.de/Publications/2019/schuster_19_rtas.pdf.

[23] Peter Wägemann, Christian Dietrich, Tobias Distler, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Whole-System Worst-Case Energy-Consumption Analysis for Energy-Constrained Real-Time Systems.“ In: *Proceedings of the 30th Euromicro Conference on Real-Time Systems (ECRTS '18)*. Volume 106. ***Outstanding Paper Award***. 2018, 24:1–24:25. DOI: 10.4230/LIPIcs.ECRTS.2018.24. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8979.

[24] Peter Wägemann, Tobias Distler, Christian Eichler, and Wolfgang Schröder-Preikschat. „Benchmark Generation for Timing Analysis.“ In: *Proceedings of the 23rd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*. 2017, pages 319–330. DOI: 10.1109/RTAS.2017.6. URL: https://www4.cs.fau.de/Publications/2017/waegemann_17_rtas.pdf.

[25] V. Sieh, R. Burlacu, T. Hönig, H. Janker, P. Raffeck, Peter Wägemann, and W. Schröder-Preikschat. „An End-to-End Toolchain: From Automated Cost Modeling to Static WCET and WCEC Analysis.“ In: *Proceedings of the 20th International Symposium on Real-Time Distributed Computing (ISORC '17)*. ***Best Paper Award***. 2017, pages 1–10. DOI: 10.1109/ISORC.2017.10. URL: https://www4.cs.fau.de/Publications/2017/sieh_17_isorc.pdf.

[26] Christian Dietrich, Peter Wägemann, Peter Ulbrich, and Daniel Lohmann. „SysWCET: Whole-System Response-Time Analysis for Fixed-Priority Real-Time Systems.“ In: *Proceedings of the 23rd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*. ***Outstanding Paper Award***. 2017, pages 37–48. DOI: 10.1109/RTAS.2017.37. URL: https://www4.cs.fau.de/Publications/2017/dietrich_17_rtas.pdf.

[27] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, and Volkmar Sieh. „A Kernel for Energy-Neutral Real-Time Systems with Mixed Criticalities.“ In: *Proceedings of the 22nd Real-Time and Embedded Technology and Applications Symposium (RTAS '16)*. 2016, pages 25–36. DOI: 10.1109/RTAS.2016.7461320. URL: https://www4.cs.fau.de/Publications/2016/waegemann_16_rtas.pdf.

[28] Peter Wägemann, Tobias Distler, Timo Hönig, Heiko Janker, Rüdiger Kapitza, and Wolfgang Schröder-Preikschat. „Worst-Case Energy Consumption Analysis for Energy-Constrained Embedded Systems.“ In: *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS '15)*. 2015, pages 105–114. DOI: 10.1109/ECRTS.2015.17. URL: https://www4.cs.fau.de/Publications/2015/waegemann_15_ecrts.pdf.

[29] Peter Wägemann. „Energieverbrauchsanalyse mittels impliziter Pfadaufzählung und genetischer Algorithmen.“ In: *Betriebssysteme und Echtzeit*. Proceedings of the Conference Echtzeit '15. 2015, pages 109–118. DOI: 10.1007/978-3-662-48611-5_12.

[30] Timo Hönig, Christopher Eibel, Benedict Herzog, Heiko Janker, Peter Wägemann, and Wolfgang Schröder-Preikschat. „Playing Hare and Tortoise: The FigarOS Kernel for Fine-Grained System-Level Energy Optimizations.“ In: *Proceedings of the 5th Brazilian Symposium on Computing Systems Engineering (SBESC '15)*. 2015, pages 80–83. URL: https://www4.cs.fau.de/Publications/2015/hoenig_15_sbesc.pdf.

Journals

[4] Kilian Müller, Johannes Weidner, Norman Franchi, and Peter Wägemann. „TinyEP: TinyML-enhanced Energy Profiling for Extreme Edge Devices.“ In: *IEEE Access* 12 (2024), pages 193747–193762. DOI: 10.1109/ACCESS.2024.3520089. URL: https://doi.org/10.1109/ACCESS.2024.3520089.

[31] Simon Ripperger, Gerald Carter, Rachel Page, Niklas Duda, Alexander Kölpin, Robert Weigel, Markus Hartmann, Thorsten Nowak, Jörn Thielecke, Michael Schadhauser, Jörg Robert, Sebastian Herbst, Klaus Meyer-Wegener, Peter Wägemann, Wolfgang Schröder-Preikschat, Björn Cassens, Rüdiger Kapitza, Falko Dressler, and Frieder Mayer. „Thinking small: next-generation sensor networks close the size gap in vertebrate biologging.“ In: *PLOS Biology* 18.4 (2020), pages 1–25. DOI: 10.1371/journal.pbio.3000655. URL: https://doi.org/10.1371/journal.pbio.3000655.

[32] V. Sieh, R. Burlacu, T. Hönig, H. Janker, P. Raffeck, Peter Wägemann, and W. Schröder-Preikschat. „Combining Automated Measurement-Based Cost Modeling with Static Worst-Case Execution-Time and Energy-Consumption Analyses.“ In: *IEEE Embedded Systems Letters* 11.2 (2019), pages 38–41. DOI: 10.1109/LES.2018.2868823. URL: https://www4.cs.fau.de/Publications/2019/sieh_19_esl.pdf.

[33] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, Volkmar Sieh, and Wolfgang Schröder-Preikschat. „Operating Energy-Neutral Real-Time Systems.“ In: *ACM Transactions on Embedded Computing Systems (ACM TECS)* 17.1 (2018), 11:1–11:25. DOI: 10.1145/3078631. URL: https://www4.cs.fau.de/Publications/2017/waegemann_17_tecs.pdf.

[34] Niklas Duda, Thorsten Nowak, Markus Hartmann, Michael Schadhauser, Björn Cassens, Peter Wägemann, Muhammad Nabeel, Simon Ripperger, Sebastian Herbst, Klaus Meyer-Wegener, Frieder Mayer, Falko Dressler, Wolfgang Schröder-Preikschat, Rüdiger Kapitza, Jörg Robert, Jörn Thielecke, Robert Weigel, and Alexander Koelpin. „BATS: Adaptive Ultra Low Power Sensor Network for Animal Tracking.“ In: *Sensors* 18.10 (2018), pages 1–34. DOI: 10.3390/s18103343. URL: https://doi.org/10.3390/s18103343.

Workshops & Work-In-Progress (peer-reviewed)

[2] Ishwar Mudraje, Jasper Devreker, Kai Vogelgesang, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, and Thorsten Herfet. „Reverse Engineering the ESP32-C3 Wi-Fi Drivers for Static Worst-Case Analysis of Intermittently-Powered Systems.“ In: *Proceedings of the 13th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems (ENSsys '25)*. 2025. DOI: 10.1145/3722572.3727926. URL: https://dl.acm.org/doi/10.1145/3722572.3727926.

[3] Emad Jacob Maroun, Eva Dengler, Christian Dietrich, Stefan Hepp, Henriette Herzog, Benedikt Huber, Jens Knoop, Daniel Wiltsche-Prokesch, Peter Puschner, Phillip Raffeck, Martin Schoeberl, Simon Schuster, and Peter Wägemann. „The Platin Multi-Target Worst-Case Analysis Tool.“ In: *Proceedings of the 22nd International Workshop on Worst-Case Execution Time Analysis (WCET '24)*. 2024. URL: https://drops.dagstuhl.de/storage/01oasics/oasics-vol121-wcet2024/OASIcs.WCET.2024.2/OASIcs.WCET.2024.2.pdf.

[12] Phillip Raffeck, Sven Posner, and Peter Wägemann. „CO2CoDe: Towards Carbon-Aware Hardware/Software Co-Design for Intermittently-Powered Embedded Systems.“ In: *Proceedings of the 3rd HotCarbon Workshop on Sustainable Computer Systems (HotCarbon '24)*. 2024. DOI: 10.1145/3727200.3727219. URL: https://sys.cs.fau.de/publications/2024/raffeck_24_hotcarbon.pdf.

[35] Sebastian Maier, Timo Hönig, Peter Wägemann, and Wolfgang Schröder-Preikschat. „Asynchronous Abstract Machines: Anti-noise System Software for Many-core Processors.“ In: *Proceedings of the 9th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS '19)*. 2019, pages 16–26. DOI: 10.1145/3322789.3328744. URL: https://www4.cs.fau.de/Publications/2019/maier_19_ross.pdf.

[36] Phillip Raffeck, Christian Eichler, Peter Wägemann, and Wolfgang Schröder-Preikschat. „Worst-Case Energy-Consumption Analysis by Microarchitecture-Aware Timing Analysis for Device-Driven Cyber-Physical Systems.“ In: *Proceedings of the 19th International Workshop on Worst-Case Execution Time Analysis (WCET '19)*. 2019, 6:1–6:12. DOI: 10.4230/OASIcs.WCET.2019.4. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10769.

[37] Christian Eichler, Peter Wägemann, and Wolfgang Schröder-Preikschat. „GenEE: A Benchmark Generator for Static Analysis Tools of Energy-Constrained Cyber-Physical Systems.“ In: *Proceedings of the 2nd Workshop on Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench '19)*. 2019. DOI: 10.1145/3312480.3313170. URL: https://www4.cs.fau.de/Publications/2019/eichler_19_cpsiotbench.pdf.

[38] Christian Eichler, Tobias Distler, Peter Ulbrich, Peter Wägemann, and Wolfgang Schröder-Preikschat. „TASKers: A Whole-System Generator for Benchmarking Real-Time-System Analyses.“ In: *Proceedings of the 18th International Workshop on Worst-Case Execution Time Analysis (WCET '18)*. 2018, 6:1–6:12. ISBN: 978-3-95977-073-6. DOI: 10.4230/OASIcs.WCET.2018.6. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9752.

[39] Heiko Falk, Sebastian Altmeyer, Peter Hellinckx, Björn Lisper, Wolfgang Puffitsch, Christine Rochange, Martin Schoeberl, Rasmus Bo Sørensen, Peter Wägemann, and Simon Wegener. „TACLeBench: A Benchmark Collection to Support Worst-Case Execution Time Research.“ In: *Proceedings of the 16th International Workshop on Worst-Case Execution Time Analysis (WCET '16)*. 2016, pages 1–10. ISBN: 978-3-95977-025-5. DOI: 10.4230/OASIcs.WCET.2016.2. URL: http://drops.dagstuhl.de/opus/volltexte/2016/6895.

[40] Peter Wägemann, Tobias Distler, Timo Hönig, Volkmar Sieh, and Wolfgang Schröder-Preikschat. „GenE: A Benchmark Generator for WCET Analysis.“ In: *Proceedings of the 15th International Workshop on Worst-Case Execution Time Analysis (WCET '15)*. Volume 47. Dagstuhl, Germany, 2015, pages 33–43. ISBN: 978-3-939897-95-8. DOI: 10.4230/OASIcs.WCET.2015.33. URL: http://drops.dagstuhl.de/opus/volltexte/2015/5254.

[41] Kai Vogelgesang, Phillip Raffeck, Peter Wägemann, Thorsten Herfet, and Wolfgang Schröder-Preikschat. „WIP: Towards a Transactional Network Stack for Power-Failure Resilience.“ In: *Proceedings of the 21st IEEE Consumer Communications & Networking Conference (CCNC WiP '24) - Work-In-Progress*. 2024. DOI: 10.1109/CCNC51664.2024.10454781. URL: https://sys.cs.fau.de/publications/2024/vogelgesang_24_ccnc_wip.pdf.

[42] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Towards System-Wide Timing Analysis of Real-Time–Capable Operating Systems.“ In: *Proceedings of the 30th Euromicro Conference on Real-Time Systems Work-in-Progress Session (ECRTS '18 WiP)*. 2018, pages 1–3. URL: https://www4.cs.fau.de/Publications/2018/schuster_18_ecrts-wip.pdf.

[43] Peter Wägemann, Tobias Distler, Phillip Raffeck, and Wolfgang Schröder-Preikschat. „Poster Abstract: Towards Code Metrics for Benchmarking Timing Analysis.“ In: *Proceedings of the 37th Real-Time Systems Symposium (RTSS '16)*. 2016, page 369. DOI: 10.1109/RTSS.2016.048. URL: https://www4.cs.fau.de/Publications/2016/waegemann_16_rtss.pdf.

[44] Peter Wägemann, Tobias Distler, Phillip Raffeck, and Wolfgang Schröder-Preikschat. „Towards Code Metrics for Benchmarking Timing Analysis.“ In: *Proceedings of the 37th Real-Time Systems Symposium Work-in-Progress Session (RTSS WiP '16)*. 2016. URL: https://www4.cs.fau.de/Publications/2016/waegemann_16_rtss-wip.pdf.

Artifact Evaluations (peer-reviewed & DOI-indexed)

[13] Markus E. Gerber, Luis Gerhorst, Ishwar Mudraje, Kai Vogelgesang, Thorsten Herfet, and Peter Wägemann. „Artifact: vNV-Heap: An Ownership-based Virtually Non-Volatile Heap for Embedded Systems.“ In: *Peer-Reviewed Artifact Evaluation, 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2025)*. 2025. DOI: 10.1145/3712433.

[14] Eva Dengler and Peter Wägemann. „Crêpe: Clock Reconfigurability for Preemption Control (Artifact).“ In: *Dagstuhl Artifacts Series* 10.1 (2024), 2:1–2:3. ISSN: 2509-8195. DOI: 10.4230/DARTS.10.1.2. URL: https://drops.dagstuhl.de/entities/document/10.4230/DARTS.10.1.2.

[15] Eva Dengler, Phillip Raffeck, Simon Schuster, and Peter Wägemann. „FusionClock: WCEC-Optimal Clock-Tree Reconfigurations (Artifact).“ In: *Dagstuhl Artifacts Series* 9.1 (2023), 2:1–2:3. ISSN: 2509-8195. DOI: 10.4230/DARTS.9.1.2. URL: https://drops.dagstuhl.de/entities/document/10.4230/DARTS.9.1.2.

[45] Simon Schuster, Peter Wägemann, Peter Ulbrich, and Wolfgang Schröder-Preikschat. *PragMetis Source Code*. Source code of the related LCTES publication titled: Annotate Once - Analyze Anywhere: Context-Aware WCET Analysis by User-Defined Abstractions. Apr. 2021. DOI: 10.5281/zenodo.4697392. URL: https://doi.org/10.5281/zenodo.4697392.

[46] Peter Wägemann, Christian Dietrich, Tobias Distler, Peter Ulbrich, and Wolfgang Schröder-Preikschat. „Whole-System WCEC Analysis for Energy-Constrained Real-Time Systems (Artifact).“ In: *Dagstuhl Artifacts Series* 4.2 (2018), 7:1–7:4. DOI: 10.4230/DARTS.4.2.7. URL: http://drops.dagstuhl.de/opus/volltexte/2018/8975.

While not necessarily being DOI-indexed, the following papers have peer-reviewed artifact evaluations:[11, 5, 8, 7, 6, 19, 20, 22, 23]

Technical Reports, Posters, & Citable Code (not peer-reviewed)

[47] Eva Dengler, Tobias Häberlein, Phillip Raffeck, and Peter Wägemann. „Positional Paper: A Clock-Distribution Abstraction for Resource-Constrained Real-Time Systems in Zephyr.“ In: *Proceedings of the 1st International Conference on Zephyr in Science and Education (ZiSE '25)*. 2025. URL: https://sys.cs.fau.de/publications/2025/2025-dengler-zephyr-sceduconf.pdf.

[48] Bernhard Heinloth, Peter Wägemann, and Wolfgang Schröder-Preikschat. „LUCI – Loader-based Dynamic Software Updates for Off-the-shelf Shared Objects.“ In: *Poster Session of the 2023 USENIX Annual Technical Conference (USENIX ATC '23)*. Poster. 2023, page 1. URL: https://sys.cs.fau.de/publications/2023/heinloth_23_atc-poster.pdf.

[49] Phillip Raffeck, Johannes Maier, and Peter Wägemann. *WoCA Source Code: Hard- & Software Prototype*. 2025. DOI: 10.5281/zenodo.14811325. URL: https://zenodo.org/records/14811326.

[50] Phillip Raffeck and Peter Wägemann. *Ecology-Aware Material Use as a Pervasive Trait in Intermittent Real-Time Systems*. Poster Session/Real-Time Pitches Session of the 35th Euromicro Conference on Real-Time Systems (ECRTS '23). Abstract: https://sys.cs.fau.de/publications/2023/raffeck_23_ecrts-abstract.pdf & Poster: https://sys.cs.fau.de/publications/2023/raffeck_23_ecrts-poster.pdf. Vienna, Austria, 2023.

[51] Alwin Berger, Simon Schuster, Peter Wägemann, and Peter Ulbrich. *OS-State–Aware Fuzzing for Worst-Case Response Times*. Abstract: https://www.betriebssysteme.org/wp-content/uploads/2022/09/FGBS2022-autumn_paper_343.pdf, Talk: https://www.betriebssysteme.org/wp-content/uploads/2022/09/FGBS2022-autumn_slides_343.pdf. Fachgruppentreffen Erlangen, Fachgruppe Betriebssysteme der Gesellschaft für Informatik (GI SYS, FG BS), September 2022. 2022.

[52] Peter Wägemann and Simon Ripperger. *Software: Ground node design for BATS tracking system*. https://doi.naturkundemuseum.berlin/data/10.7479/z5ym-kx58. 2020. DOI: 10.7479/z5ym-kx58.

[53] Peter Wägemann, Florian Harbecke, Bernhard Heinloth, Henriette Hofmeier, and Wolfgang Schröder-Preikschat. *An Energy-Neutral, WiFi-Connected Room Display with Hand-Crank–Based Energy Harvesting*. Implementation Award, 2nd place, FAU Ideenwettbewerb (FAU idea competition), proposal: https://www4.cs.fau.de/Publications/2019/waegemann_19_fau_ideenwettbewerb.pdf. 2019.

[54] Christian Dietrich and Peter Wägemann. „SysWCET: Ende-zu-Ende-Antwortzeiten für OSEK-Systeme.“ In: *Proceedings of the Embedded Software Engineering Kongress (ESE '17)*. 2017. URL: https://sra.uni-hannover.de/Publications/2017/dietrich_17_ese.pdf.

[55] Christian Eichler, Peter Wägemann, Tobias Distler, and Wolfgang Schröder-Preikschat. „Demo Abstract: Tooling Support for Benchmarking Timing Analysis.“ In: *Proceedings of the 23rd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*. 2017, pages 159–160. DOI: 10.1109/RTAS.2017.20. URL: https://www4.cs.fau.de/Publications/2017/eichler_17_rtas-demo.pdf.

[56] Christopher Eibel, Sebastian Herbst, Björn Cassens, Timo Hönig, Peter Wägemann, Heiko Janker, Rüdiger Kapitza, Klaus Meyer-Wegener, and Wolfgang Schröder-Preikschat. *A Flexible, Adaptive System for Data-Stream Processing in Energy-Constrained Ad-hoc Networks*. Technical report. Erlangen: Friedrich-Alexander University Erlangen-Nürnberg (FAU), 2016, pages 1–8. URL: https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/6893.

[57] Peter Wägemann, Timo Hönig, Rüdiger Kapitza, and Wolfgang Schröder-Preikschat. *Worst-Case Energy Consumption Analysis for Soft and Hard Energy Systems*. Poster Session at the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14). 2014.

Dissertation

[58] Peter Wägemann. „Energy-Constrained Real-Time Systems and Their Worst-Case Analyses." PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2020. URL: https://nbn-resolving.org/urn:nbn:de:bvb:29-opus4-146935.

Dissertation-Related Awards

[59] Peter Wägemann. „Static Worst-Case Analyses and Their Validation Techniques for Safety-Critical Systems." In: *Ernst Denert Award for Software Engineering 2020*. 2022, pages 227–247. URL: https://doi.org/10.1007/978-3-030-83128-8_11.

[60] Peter Wägemann. „Energiebeschränkte Echtzeitsysteme und ihre Worst-Case-Analysen." In: *Ausgezeichnete Informatikdissertationen 2020*. Gesellschaft für Informatik e.V., 2021, pages 329–338. URL: http://dl.gi.de/handle/20.500.12116/37920.

Edited Books

[61] Peter Wägemann, editor. *21st International Workshop on Worst-Case Execution Time Analysis (WCET 2023)*. Volume 114. OASIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. ISBN: 978-3-95977-293-8. URL: https://drops.dagstuhl.de/entities/volume/OASIcs-volume-114.

[62] Doğanalp Ergenç, Shadi Attarha, and Peter Wägemann, editors. *Proceedings of the 1st Workshop on Resilient Networks and Systems (ReNeSys 2025)*. Berlin, Germany, 2025. DOI: 10.14279/depositonce-24399.

Supervised Doctoral/PhD Theses

[63] Phillip Raffeck. „Predictable Execution for Device-Bound Embedded Systems with Intermittent Power." PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2025. DOI: 10.25593/open-fau-2372. URL: https://doi.org/10.25593/open-fau-2372.

APPENDIX B
PAPER REPRINTS

The 12 peer-reviewed publications, listed as follows, represent the central parts of this cumulative habilitation treatise. These publications are provided as personal reprints at the end of this document. All copyrights remain with the authors and/or the respective publishers (i.e., ACM, IEEE, Dagstuhl Publishing). Links to the PDF files of these papers are also available on my personal website (as authors' version).

Chapter 3: Static & Dynamic Program Analysis

LCTES '24 Phillip Raffeck, Johannes Maier, and Peter Wägemann [1]
WoCA: Avoiding Intermittent Execution in Embedded Systems by Worst-Case Analyses with Device States
Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '24)

ENSsys '25 Ishwar Mudraje, Jasper Devreker, Kai Vogelgesang, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, and Thorsten Herfet [2]
Reverse Engineering the ESP32-C3 Wi-Fi Drivers for Static Worst-Case Analysis of Intermittently-Powered Systems
Proceedings of the 13th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems (ENSsys '25)

WCET '24 Emad Jacob Maroun, Eva Dengler, Christian Dietrich, Stefan Hepp, Henriette Herzog, Benedikt Huber, Jens Knoop, Daniel Wiltsche-Prokesch, Peter Puschner, Phillip Raffeck, Martin Schoeberl, Simon Schuster, and Peter Wägemann [3]
The Platin Multi-Target Worst-Case Analysis Tool
Proceedings of the 22nd International Workshop on Worst-Case Execution Time Analysis (WCET '24)

Access '24 Kilian Müller, Johannes Weidner, Norman Franchi, and Peter Wägemann [4]
TinyEP: TinyML-enhanced Energy Profiling for Extreme Edge Devices
IEEE Access

RTSS '25 Alwin Berger, Simon Schuster, Peter Wägemann, and Peter Ulbrich [5]
Dynamic Fuzzing-Based Whole-System Timing Analysis
Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)

Chapter 4: Optimizing Resource Demands

ECRTS '23 Eva Dengler, Phillip Raffeck, Simon Schuster, and Peter Wägemann [6]
FusionClock: Energy-Optimal Clock-Tree Reconfigurations for Energy-Constrained Real-Time Systems
Proceedings of the 35th Euromicro Conference on Real-Time Systems (ECRTS '23)

ECRTS '24 Eva Dengler and Peter Wägemann [7]
Crêpe: Clock-Reconfiguration–Aware Preemption Control in Real-Time Systems with Devices
Proceedings of the 36th Euromicro Conference on Real-Time Systems (ECRTS '24)

Chapter 5: Utilizing Analyses & Optimizations for Systems

LCTES '25 Markus E. Gerber, Luis Gerhorst, Ishwar Mudraje, Kai Vogelgesang, Thorsten Herfet, and Peter Wägemann [8]
vNV-Heap: An Ownership-based Virtually Non-Volatile Heap for Embedded Systems
Proceedings of the 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2025)

CCNC '25 Kai Vogelgesang, Ishwar Mudraje, Luis Gerhorst, Phillip Raffeck, Peter Wägemann, Thorsten Herfet, and Wolfgang Schröder-Preikschat [9]
PfIP: A UDP/IP Transactional Network Stack for Power-Failure Resilience in Embedded Systems
Proceedings of the 22nd IEEE Consumer Communications & Networking Conference (CCNC 2025)

RTAS '24 Harald Böhm, Tobias Distler, and Peter Wägemann [10]
TinyBFT: Byzantine Fault-Tolerant Replication for Highly Resource-Constrained Embedded Systems
Proceedings of the 30th Real-Time and Embedded Technology and Applications Symposium (RTAS '24)

RTSS '25 Tobias Häberlein, Eva Dengler, Phillip Raffeck, and Peter Wägemann [11]
WatwaOS: A Framework for Worst-Case–Aware Tailoring and Whole-System Analysis of Energy-Constrained Real-Time Systems
Proceedings of the 46th IEEE Real-Time Systems Symposium (RTSS '25)

HotCarbon '24 Phillip Raffeck, Sven Posner, and Peter Wägemann [12]
CO2CoDe: Towards Carbon-Aware Hardware/Software Co-Design for Intermittently-Powered Embedded Systems
Proceedings of the 3rd HotCarbon Workshop on Sustainable Computer Systems (HotCarbon '24)

[The paper reprints have been excluded from this PDF. These papers are publicly accessible through the links in the bibliography in Appendix A.]