Whole-System WCEC Analysis for Energy-Constrained Real-Time Systems (Artifact)*

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

Christian Dietrich
Leibniz Universität Hannover (LUH), Germany

Tobias Distler
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

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

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

Abstract

Although internal devices (e.g., memory, timers) and external devices (e.g., sensors, transceivers) significantly contribute to the energy consumption of an embedded real-time system, their impact on the worst-case response energy consumption (WCRE) of tasks is usually not adequately taken into account. Most WCRE analysis techniques only focus on the processor and neglect the energy consumption of other hardware units that are temporarily activated and deactivated in the system.

To solve the problem of system-wide energy-consumption analysis, we present SysWCEC, an approach that addresses these problems by enabling static WCRE analysis for entire real-time systems, including internal as well as external devices. For this purpose, SysWCEC introduces a novel abstraction, the power-state–transition graph, which contains information about the worst-case energy consumption of all possible execution paths. To construct the graph, SysWCEC decomposes the analyzed real-time system into blocks during which the set of active devices in the system does not change and is consequently able to precisely handle devices being dynamically activated or deactivated.

In this artifact evaluation, which accompanies our related conference paper, we present easy to reproduce WCRE analyses with the SysWCEC framework using several benchmarks. The artifact comprises the generation of the power-state–transition graph from a given benchmark system and the formulation of an integer linear program whose solution eventually yields safe WCRE bounds.

2012 ACM Subject Classification Computer systems organization → Real-time systems

Keywords and phrases energy-constrained real-time systems, worst-case energy consumption (WCEC), worst-case response energy consumption (WCRE), static whole-system analysis

Digital Object Identifier 10.4230/DARTS.4.2.7

Related Article 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 2018), LIPIcs, Vol. 106, pp. 24:1–24:24 , 2018.
http://dx.doi.org/10.4230/LIPIcs.ECRTS.2018.24

Related Conference 30th Euromicro Conference on Real-Time Systems (ECRTS 2018), July 3–6, 2018, Barcelona, Spain

* This work is supported by the German Research Foundation (DFG), in part by Research Unit FOR 1508 under grant no. SCHR 603/14-2, Research Grant no. SCHR 603/13-1, the CRC/TRR 89 Project C1 (Invasive Computing), and the Bavarian Ministry of State for Economics under grant no. 0704/883 25.

1 Scope

In the following, we first give an overview of the main problems when statically analyzing system-wide worst-case response energy consumption (WCRE)[1] of energy-constrained real-time systems and outline our *SysWCEC* approach to tackle these problems (see Section 1.1). Subsequently, we present the workflow of *SysWCEC*'s artifact evaluation (see Section 1.2).

1.1 Whole-System Energy-Consumption Analysis

Energy-constrained real-time systems usually use several internal devices, such as timers, or external peripherals, such as sensors or transceivers, to interact with the environment. In addition to providing timeliness by using analyzed worst-case execution-time bounds, energy-constrained real-time systems need to further guarantee that tasks execute within predefined energy-budget bounds [11, 14]. When these systems are operated under the common preemptive, fixed-priority scheduling strategy, a task, for example, with lower priority, which temporarily activates a device and thereby increases the system's overall power consumption, can be preempted by a task with higher priority. As a consequence, in order to statically provide sound bounds on the energy consumption, determining the worst-case energy consumption (WCEC) of both tasks in isolation is no longer sufficient, due to their mutual influence on each other: The low-priority task's response energy consumption is increased by the intermediate execution of the high-priority task and the high-priority task's response energy consumption is influenced if the lower task turns on a device prior to being preempted. In other words, the WCRE analysis needs to consider all preemptions and device (de-)activations along the feasible program paths of the whole real-time system.

A sound approach to address devices for WCRE analysis is to assume that all devices are active during the entire runtime. However, this approach leads to overly pessimistic results in the light of the fact that peripherals are usually switched on for the short timespan when their service is actually required. In order to solve this problem of pessimistic WCRE estimates, we decompose the system's source code into blocks where devices are either active or inactive. For the system-wide path analysis we use the `dOSEK` [2] framework, which is both a system analyzer and an operating-system generator for OSEK-compliant real-time systems. `dOSEK` is able to explicitly enumerate all possible system paths and thereby considers possible preemptions by asynchronous interrupts, synchronous task activations, and the fixed-priority scheduling semantic. We extended `dOSEK` to make use of the blocks with temporarily activated devices. After `dOSEK`'s system-path enumeration under consideration of the devices' power state, we obtain an abstraction that now enables system-wide WCRE analyses: the *power-state–transition graph*, or *PSTG* for short.

With the PSTG storing knowledge of all possible system-wide program paths and the active devices along these paths, we are able to formulate an integer linear program (ILP). This approach of finding the maximum flow through a directed graph is similar to the well-known implicit path-enumeration technique [6, 9]. Additionally, *SysWCEC* is able to safely consider the scheduling semantic and asynchronous interrupts. In our evaluation, we extended the `Platin` worst-case analysis toolkit [8] for the formulation of ILP. Solving the formulated ILP problem by means of specialized solvers (e.g., `lp_solve`) eventually yields bounds on the WCRE.

Our implementation relies on several modified frameworks [2, 8, 10]. Furthermore, for the implementation of *SysWCEC*, we benefited from our previous work on response-time analysis [4] and infrastructure around our benchmark generator for worst-case execution-time analysis [5, 13].

[1] We use the terms WCEC and WCRE analogous to timing analysis where the worst-case execution time (WCET) refers to a task in isolation and the worst-case response time (WCRT) to the timespan from the start of a task until its completion, including all possible interferences (e.g., preemptions).

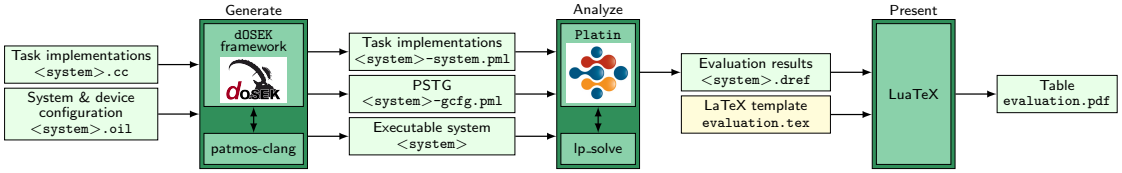

Figure 1 Workflow of *SysWCEC*'s artifact evaluation

1.2 Artifact-Evaluation Workflow

The main goal of our artifact evaluation is the reproduction of evaluation results and the visual comparison against published values with a single command. Figure 1 gives an overview of the artifact evaluation's workflow. We consider several benchmarks in the evaluation, denoted as `system` in the figure. The system's source code together with the configuration (e.g., containing information on the devices' power consumption) is input to the `dOSEK` framework, which analyzes the system. When executing the command, the `dOSEK` framework first generates the analyzed system (including the power-state–transition graph) from the given task implementations and configurations. `dOSEK`'s output is input to the modified version of the `Platin` analysis toolkit. It formulates the integer linear program, which is solved by `lp_solve`. In the last processing step of the evaluation, `Platin`'s analysis results, which are stored using `dataref` [3], and a LaTeX template are input to the `LuaTeX` typesetter, which produces a final PDF document containing the evaluation results. This document can be directly compared to the published values.

2 Content

The artifact package consists of a virtual-machine image (Lubuntu 16.04, amd64) with all required software installed. The most importantly required software for *SysWCEC* are partly modified versions of the following programs:

- `Patmos`' `LLVM`/`Clang` compiler infrastructure [10]
- `dOSEK` system analyzer/generator [2]
- `Platin` worst-case analyses toolkit [8]
- `lp_solve` ILP solver [1]

3 Getting the Artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition to this, the artifact's code is available at *SysWCEC*'s code-repository website:

`https://gitlab.cs.fau.de/syswcec/`

The main repository of the artifact evaluation (`ecrts18-artifact-evaluation/`), also contains a link to the (latest) virtual machine with the installed programs. For updates, especially on our future work in the context of *SysWCEC*, such as state-enumeration and ILP-solving enhancements (for details see our conference paper [12]), please update the related source-code repositories (i.e., `dOSEK`, `Platin`, the artifact repository itself). The artifact's repository contains information on how to execute the commands to follow the workflow (see Section 1.2) in order to reproduce the provided evaluation results (see `README.md`).

4 Tested Platforms

To enable a high degree of portability, we installed the required software inside a virtual-machine image, which can be executed using the `VirtualBox` hypervisor [7]. Although the size of the artifact requires around 4 GiB of disk space, the self-contained virtual-machine image avoids the necessity to install and configure the multitude of required tools (e.g., `Patmos`' `Clang`, `dOSEK`, `Platin`). `VirtualBox` is available for all common platforms.

5 License

GNU General Public License Version 3

6 MD5 Sum of the Artifact

b0c7fd4f2e18757d7fd06c92c40cef75

7 Size of the Artifact

4.2 GiB

References

1 Michel Berkelaar, Kjell Eikland, and Peter Notebaert. lp_solve – mixed integer linear programming (MILP) solver. `lpsolve.sourceforge.net`.

2 Christian Dietrich, Martin Hoffmann, and Daniel Lohmann. Global optimization of fixed-priority real-time systems by RTOS-aware control-flow analysis. *ACM Transactions on Embedded Computing Systems (ACM TECS)*, 16:35:1–35:25, 2017.

3 Christian Dietrich and Daniel Lohmann. The dataref versuchung: Saving time through better internal repeatability. *ACM SIGOPS Operating Systems Review*, 49(1):51–60, 2015.

4 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 23nd Real-Time and Embedded Technology and Applications Symposium (RTAS '17)*, pages 37–48, 2017.

5 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)*, pages 159–160, 2017.

6 Yau-Tsun Steven Li and Sharad Malik. Performance analysis of embedded software using implicit path enumeration. In *ACM SIGPLAN Notices*, volume 30, pages 88–98, 1995.

7 Oracle Corporation. VirtualBox. `virtualbox.org`.

8 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 9th Workshop on Software Technologies for Future Embedded and Ubiquitious Systems (SEUS '13)*, pages 33–40, 2013.

9 Peter Puschner and Anton Schedl. Computing maximum task execution times: A graph-based approach. *Real-Time Systems*, 13:67–91, 1997.

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

11 Marcus Völp, Marcus Hähnel, and Adam Lackorzynski. Has energy surpassed timeliness? – scheduling energy-constrained mixed-criticality systems. In *Proceedings of the 20th Real-Time and Embedded Technology and Applications Symposium (RTAS '14)*, pages 275–284, 2014.

12 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)*, pages 1–24, 2018.

13 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)*, pages 319–330, 2017.

14 Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, Volkmar Sieh, and Wolfgang Schröder-Preikschat. Operating energy-neutral real-time systems. *ACM Transactions on Embedded Computing Systems (ACM TECS)*, 17(1):11:1–11:25, 2017.