Supporting Development of Energy-Optimised Java Real-Time Systems using TetaSARTS
Luckow, Kasper Søe; Bøgholm, Thomas; Thomsen, Bent

Publication date:
2013

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

? Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
? You may not further distribute the material or use it for any profit-making activity or commercial gain
? You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.
Motivation

It is well-known, that the traditional Java run-time is unsuited for use in embedded real-time systems, which is attributed issues such as the lack of high-resolution real-time clocks and timers, insufficient thread semantics, and, most notably, memory management, which is traditionally handled by a garbage collector whose execution is highly unpredictable. However, with emerging standards such as the Real-Time Specification for Java (RTSJ) and the Safety Critical Java (SCJ) profile, these issues have been accounted for, thereby achieving a significant step towards use in embedded real-time systems development.

Having a suitable programming model as introduced by e.g. SCJ and RTSJ is not the only component in making Java a viable technological alternative and competitor to C in the embedded real-time systems domain. Equally important is the complementation of tools and analyses that support the development, and for verification purposes. For real-time systems, the latter comprises functional correctness, but also temporal correctness, which is the focus of this work.

Contributions

The TetraSARTS tool whose capabilities include:

Schedulability Analysis The analysis is approached as a reachability, model checking problem by determining whether the Timed Automata model satisfies the specification All not deadlock. A deadlock can only be reached, if there exists a simulation reaching the DeadlineErrorRun location for any of the real-time tasks (see Fig. 1).

Clock Frequency Adjustment The analysis facilitates adjustment of the clock frequency. Using a technique such as the bisection method, the appropriate clock frequency can be determined while still ensuring schedulability. Thus, potential energy reductions can be identified.

Worst-Case Response Time Analysis The Timed Automata model facilitates Worst-Case Response Time (WCRT) analysis to be conducted taking into account blockage timing. Using the sup-query extension of the UPPAAL specification language, the maximum observed value of a clock variable can be found. Hence, e.g. for periodic tasks, WCRT analysis can be conducted using the specification sup(periodicThread.ExecutingTask): periodicThread.wcrt (see Fig. 1).

Processor Utilisation/Idle Time Analysis Adding the Timed Automaton shown in Fig. 2 allows monitoring the utilisation and idle time of the processor using the specification sup: util, idle.

TetraSARTS

TetraSARTS is a fully automated tool for conducting timing analyses of SCJ Java Bytecode programs taking into account the execution environment (see Fig. 3 for an overview). It accommodates hardware implementations of the JVM, and traditional execution environments with a software implementation of the JVM including the hosting hardware. It supports both periodic and sporadic real-time tasks, and from these, it generates Timed Automata simulating their control flow. The structure of the model and the communication between the Timed Automata using synchronisation actions is shown in Fig. 4. The analyses are performed using the UPPAAL model checker.

Results

We base the results on analysing two representative examples of real-time systems written in Java: the Real-Time Sorting Machine (RTSM) and the Minepump control system. The systems have been evaluated on the Java Optimized Processor (JOP), and an execution environment consisting of the hardware platform Virtual Machine (HVM) on an AVR ATmega2560 microcontroller. The results have been obtained on a system with an Intel Core i7-2620M @ 2.70GHz and 8 GB of memory. The results are shown in Table 1, 2, and 3.

Future Work

Support other Analyses The Timed Automata model can be used for analysing other interesting properties e.g. worst case blocking time. Statistical Model Checking[2] Can be used for analysing systems for which probabilistic guarantees of temporal correctness are sufficient. The technique can be used to avoid exhaustive exploration of the state-space of the model.

Schedulability Abstractions[1] Decorate Java interfaces with abstract, behavioural descriptions and generate Timed Automata accordingly. Case Studies We want to further evaluate the applicability of TetraSARTS by using more complex real-time systems, and possibly other variations of the execution environment.

Download TetraSARTS

http://people.cs.aau.dk/~luckow/tetesarts/

References
