Measurement-Based Worst-Case Execution Time Analysis[*]

Ingomar Wenzel, Raimund Kirner, Bernhard Rieder, Peter Puschner
Institute of Computer Engineering, Vienna University of Technology
Real-Time Systems Group, Treitlstrasse 3, 1040 Vienna, Austria
{ingo, raimund, bernhard, peter}@vmars.tuwien.ac.at

Abstract

In the last years the number of electronic control systems has increased significantly. In order to stay competitive more and more functionality is integrated into more and more powerful and complex computer hardware. Due to these advances in control systems engineering new challenges for analyzing the timing behavior of real-time computer systems arise. The two identified main challenges are execution-time modeling of the hardware and the path problem that forbids capturing the worst-case execution time (WCET) by end-to-end measurements due to limits in computational complexity. This work presents the cornerstones of our new measurement-based WCET analysis method that successfully addresses these problems. We clearly identify our research goals and the relevance of our research. Especially, the novel aspects of our approach are emphasized. The conclusion is formed by a brief presentation of an industrial-size case study application.

1. Motivation

The last years have seen significant advances in electronic control systems replacing more and more conventional control systems. Control systems continuously interact with their environment. Thus, in order to fulfill the desired control function, these systems do not only have to perform correct calculations in the value domain, but also have to provide the computed results within specified time bounds, to react accordingly to the dynamics of the environment.

Further, driven by the increased flexibility resulting from the use of microprocessors in control systems, more and more functionality is integrated into electronic control units (ECUs) causing higher complexity of the control applications.

Additionally, a shift towards the complete replacement of mechanical backup solutions by electronic control systems takes place. This causes an increase in the number of safety-critical electronic control systems. Safety-critical means that a failure of such systems may result in catastrophic consequences.

The automotive industry is one of the leading fields pushing these developments. Today, a new car contains about 30 ECUs (in luxus cars up to 70 ECUs are embedded [5]). In the automotive industry about 55% of breakdowns can be traced back to problems in electronic systems. About 30% of these incidents are estimated to be caused by timing problems.

The worst-case execution time (WCET) is the longest time it takes to execute a given program code. The determination of the WCET is a tough topic and of tremendous importance to ensure the proper function of real-time systems.

The highly active research field of WCET analysis has brought a number of well established static WCET analysis methods. However, these static analysis methods [1] cannot keep pace with the rapidly increasing analysis complexity of modern control systems. Especially, the task of manually modeling the timing behavior of the hardware is very time consuming and error-prone.

2. Proposed Approach

Within our research project "Model-based Development of Distributed Embedded Control Systems" (MoDECS) a completely new hybrid measurement-based WCET analysis method is developed. The key strength of this approach is the fruitful combination of the advantages of static program analysis and runtime measurements of the application on real hardware [3]. In more detail, our method involves the following steps:

Static Analysis: In the first step, a static analysis of the C source code allows a concise and safe analysis of

[*] This work has been supported by the FIT-IT research project "Model-based Development of Distributed Embedded Control Systems" (MoDECS).

the overall program structure. In contrast to common methods working on object code level, this ensures a high level of portability because C is a well established programming language standard in control systems engineering. Additionally, C is also used as output format by code generation tools like *Real-Time Workshop* (Mathworks Inc.) or *TargetLink* (dSpace GmbH).

Partitioning: As the second step, our method allows to overwhelm the complexity by means of automatically decomposing the program into well manageable subparts called program segments. Figure 1 illustrates the control-flow graph decomposition resulting in the program segment graph.

Test Data Generation: Next, we want to obtain the execution times that our task spends within each of the identified program segments. Therefore, we have to exhaustively guide the task's execution into those paths that we need for acquiring timing information. Since the paths taken through the task during execution depend on the input data to the task, we have to find suitable test-data vectors enforcing exactly these paths. This problem is solved by automatic test data generation. We introduced a new three-level test data generation process. First, we use random test data vectors to cover a broad number of paths. Next, we apply an evolutionary algorithm for paths that are harder to find.

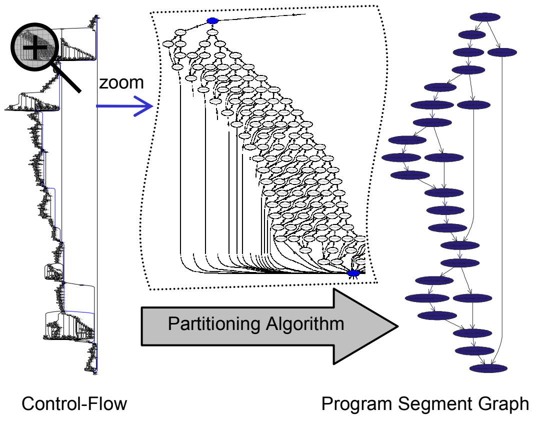

Figure 1. Control-flow graph partitioning

The great aspect of using evolutionary algorithms this way is that we definitely know if a result delivered by the algorithm is a solution to our problem. The better the results delivered by this algorithm, the fewer test cases remain to be generated by the third and last step. For generating the remaining test cases we adopted model checking for test-data generation. Using this relatively expensive method (in terms of computational resource needs) guarantees a test-data vector can be found whenever one exists. Using these three levels of data generation, our method efficiently generates the required test data.

Execution Time Measurements: The generated test-data are used to execute the calculated paths within the program segments. The timing information is captured by code instrumentations that are automatically generated and placed on block boundaries. The used modifications do not change the timing behavior or at least modify it in a predictable way.

Calculation Step: The obtained execution times are safely combined; thus yielding the overall WCET bound by a final calculation. This calculation step makes use of the structural information acquired in the first step.

3. Research Goals and Methodology

Our main research goal is to demonstrate the proof-of-concept of our approach. Before we started implementing the ideas of this new method, we performed a feasibility analysis. Now, a framework incorporating all steps of our method is being implemented. When implementing the static analysis step, a lot of pre-existing knowledge from static analysis methods can be reused. To show the proof-of-concept, a number of highly exciting questions subject to our research arise:

- Automatically parameterizable control-flow graph partitioning for complexity reduction. A basic decomposition algorithm has been implemented; finding optimizations in the partitioning algorithm is subject to our research.
- Incorporating path information in the time modeling process. Are there alternative strategies for optimizing the tightness of the WCET bound delivered by this approach?
- Development of the automated test data generation process. Will it be practicable to generate all the required test-data vectors?
- Allow schedulability analysis using the determined WCET bounds. Especially, when modeling complex hardware this question is of high impact. Is it possible to allow the safe composition of WCET analysis results at component level?
- Due to increased complexity in control systems, companies use high-level design tools like MATLAB/SIMULINK to model these systems. Then, out of these models the actual control code is automatically generated by using code generators (e.g., Real-Time Workshop, TargetLink). Unfortunately, this automatic generation leads to a high

complexity of the generated code. However, we want to investigate how to use structural features resulting from this automatic generation in order to simplify the analysis.

4. Relevance of the Approach

The currently established WCET analysis approaches are not able to cope sufficiently with the rising demands resulting from the highly dynamic developments in the field of electronic control engineering [1]. However, as outlined in the beginning, the fulfillment of the temporal constraints in safety-critical applications is a stringent imperative to be satisfied in order to avoid catastrophic consequences. When considering the current situation, there is urgent call for action.

First, a main deficiency of existing WCET methods is the high dependence on the hardware platform of the target system [2]. The execution time of all instruction types has to be modeled manually. This is a very complex, cost-intensive and error-prone task. Often, especially for every new processors, huge parts of the whole analysis framework have to be rewritten. Our method addresses this problem in an elegant way: instead of statically calculating the execution time, we guide the execution of the program and measure the execution times. These times are then combined by analytical methods to compute a safe WCET bound.

Second, existing methods are highly dependent on the deployed software tools and tool chains. A number of approaches modify the compiler source code in order to integrate the WCET analysis tool, but this turned out not to be a practical way. Additionally, every time when new versions of the compiler tool chain are released the WCET analysis tools have to be adapted. Our method works on the level of C source code and uses the unmodified tool chain to compile and execute the program to measure the execution times.

Third, our method makes use of code structures that can be found inherent in automatically generated program code since automatic code generation is of high increasingly high importance in control systems engineering [4].

5. Novelty and Originality

Besides the high importance of a practically useful WCET analysis method, we are able to contribute a number of completely new ideas in the research field. The main contributions of our approach have been identified as follows:

- The introduction of a hybrid WCET analysis approach using a measurement-based execution time model.
- Using heuristics in a deterministic manner for test data generation, i.e., we know whether the calculated results are solutions to the problem.
- We use model checking for test-data generation. Model checking usually is used for formal system validation where complex problems have to be solved efficiently. We apply this technique for solving the problem of automatic test data generation.
- We automatically perform a breakdown in the application complexity while concurrently incorporating path information into the timing model (implicit feasible path partitioning).
- We develop clear architectural constraints that allow the safe modular combination of the WCET bounds from individual tasks. In other words, we establish the requirements of stable boundaries between the WCET of tasks and their scheduling.

6. Case Study Application

In this project, a close cooperation with two leading suppliers in the automotive industry takes place (Magna Steyr Fahrzeugtechnik Graz and AVL List, both located in Graz, Austria) enabling the great opportunity to check our ideas against industrial control applications.

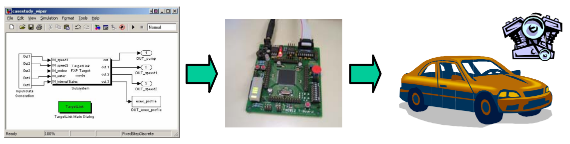

Figure 2. Case study application

We have set up equivalent tool chains and are able to use real-sized state-of-the-art models for code generation. A simplified version of such a tool chain is depicted in Figure 2. Starting from a MATLAB/SIMULINK model, a code generator is invoked to generate highly optimized C-code for the desired target platform. This C-code is analyzed by our tools. We add instrumentations to perform the required measurements. Then the compiled executable is uploaded into the target hardware where it is executed and runtime measurements take place. For the final production code, the instrumentations are removed and the code can be deployed into the power train control unit. In more detail, one real-time task subject to our investigation contains about 10^{32} paths. Our algorithm partitions this program into 25 program segments and needs a total of about 30,000 measurements (those can be captured in about 6 minutes on the HCS12 target platform) to determine the execution time model. However, due to intellectual property issues no detailed information about the code can be presented.

7. Conclusion

The presented WCET analysis approach successfully addresses the hardware modeling challenge by applying our paradigm the "best hardware model is the hardware itself". We successfully deploy this idea by using measurements for execution-time modeling. Further, we solve the path problem by automatically decomposing the program control-flow graph into well manageable units and use automatically generated test-data for guiding the program executions.

We have outlined the novelty of our approach, especially important is the new way of generating the required test-data, applying implicit feasible-path partitioning and investigating stable boundaries for composing the WCETs of tasks.

The practical strength of our method has been shown by an application of the method on automatically generated industrial-sized control applications.

References

[1] "Discussion of Misconceptions about WCET Analysis", Raimund Kirner, Peter Puschner; *Proc. 3rd Euromicro International Workshop on WCET*, 2003

[2] "The Influence of Processor Architecture on the Design and the Results of WCET Tools", Reinhold Heckman, Marc Langenbach, Stephan Thesing, Reinhard Wilhelm; *IEEE Proceedings on Real-Time Systems*, 91(7):1038–1054, 2003

[3] "Automatic Timing Model Generation by CFG Partitioning and Model Checking", Ingomar Wenzel, Bernhard Rieder, Raimund Kirner, Peter Puschner; *Design Automation and Test in Europe (DATE)*, March 2005

[4] "A Study of Automatic Code Generation for Safety-Critical Software: Preliminary Report", L. Crawford, J. Erwin, S. Grimaldi, S. Mitra, A. Kornecki, D.P. Gluch; *High Assurance Systems Engineering. Proceedings. Eighth IEEE International Symposium on*, pp. 287 – 288, 2004

[5] "The electrical/electronic diagnostic concept of the new 7 series", A. Deicke; In *Convergence International Congress & Exposition On Transportation Electronics*, Detroit, MI, USA; SAE, 2002