WCET Determination Tool for Embedded Systems Software

Carsten Albrecht, Roman Koch,
Thilo Pionteck, Erik Maehle
Institute of Computer Engineering
University of Lübeck
Ratzeburger Allee 160
23538 Lübeck, Germany
albrecht@iti.uni-luebeck.de

Michael Werner, Rudolf Fuchsen

Sysgo AG
Am Pfaffenstein 14
55270 Klein-Winternheim, Germany
michael.werner@sysgo.com

1. INTRODUCTION

Embedded systems are special-purpose computer systems within environments such as automotive, avionic, telecommunication, etc. As these systems often perform time-critical control tasks, it is necessary to precisely predict their behaviour. Software-based solutions on the one hand provide high flexibility. On the other hand, especially when multiple software processes are to be executed concurrently on the same hardware, the software approach makes prediction more difficult. Predictions essentially rely on the knowledge of when a software process terminates. For a reasonable system architecture it has to be known at an early stage of design time to scale the system well. Dynamic factors such as cache usage, amount of data, and availability of shared resources have great impact on the result. Thus, tools are necessary to determine execution time as accurate as possible. The execution time is commonly approximated by determining the interval of best and worst case execution time (BCET & WCET). While the BCET can be roughly estimated by evaluating the shortest path the achievement of good approximations for the WCET is heavy. Considering all possible execution paths of the analysed program as well as the effects of the underlying architecture on the timing properties affected by pipelining and caches create a hard issue to solve. Nevertheless these figures are important in certification processes in the application fields given above as well as performance prediction and solving design issues. Another motivation is the lack of tools targeted to an intermediate level of abstraction. In general tools are re-targetable frameworks analysing high-level code with sophisticated algorithms or similar algorithms are applied to low-level code that lacks context and type information.

2. TOOL DESCRIPTION

Existing tools such as *Cinderella* based on [2] et al. determine the WCET by ILP systems. It is re-targetable on lower-level code for different architectures. The tool sketched here is focused on the wide-spread PowerPC architecture [1] which is not supported by *Cinderella*. The PowerPC includes a super-scalar pipeline, a translation look-aside buffer (TLB), and a branch history table for branch prediction. Beside pipeline consumption and memory latencies side-effects such as cache hits and misses have to be considered for run-time prediction. As a basic analysis method implicit path enumeration (PE) as *Cinderella* does and code simulation are used. The PE algorithm applies an approach that includes caches [3]. The results of PE are in general too pessimistic because it may include unused paths or cannot bound loops. Therefore, the tool analyses the code and requests for certain programmer knowledge to enhance the results. If applicable the more accurate method of code simulation is used. It delivers more precise results as exemplarily shown for frequently used methods and algorithms such as `sign` or `bubble sort`.

The tool performs its analysis on object code including debug information. So, there is just a low abstraction of the underlying hardware but some useful information of the source code is still preserved. For PE an ILP equation system is built. It is enriched by programmer knowledge so that even formerly unsolvable equation systems become solvable. Necessary user input and value ranges of variables such as loop counters can be defined in advance or interactively on request. Dynamic impact factors such cache-miss or TLB miss penalties are parametrisable so that the analysis is re-targetable within the PowerPC family. Currently, only instruction cache analysis is implemented, data-cache evaluation based on existing techniques is in development.

3. CONCLUSIONS

The analysis tool utilises state-of-the-art techniques for WCET determination. It is targeted to a medium level of abstraction to include certain hardware features as well as source code information. In case of code simulation failures the WCET can be computed by PE with a higher degree of pessimism. All in all, the analysis tool is targeted to embedded functions with only few user interaction.

4. REFERENCES

[1] B. Frey. *PowerPC Architecture Book*. International Business Machines Corporation, Nov. 2005.

[2] Y.-T. S. Li and S. Malik. Performance analysis of embedded software using implicit path enumeration. In *Proc. of the 32nd ACM/IEEE Conference on Design Automation*, pages 456–461, 1995.

[3] J. Staschulat and R. Ernst. Worst case timing analysis of input dependent data cache behavior. In *Proc. of the 18th Euromicro Conference on Real-Time Systems*, 2006.