Timing Predictability — a Must for Avionics Systems

Reinhard Wilhelm
Saarland University
Saarbrücken, Germany

Lothar Thiele
Swiss Federal Technical Institute
Zuerich, Switzerland

Timing Analysis – the Problem

This position statement concentrates on the *certification of the timing properties* of embedded avionics systems with *hard real-time* characteristics. For *certification*, they need offline guarantees for the *satisfaction of their timing constraints*, and these should be derived by *sound methods*.

In modern microprocessor architectures, caches, pipelines, and all kinds of speculation are key features for improving (average-case) performance. They also increase the variability of execution times; an individual instruction may take (amortized) one machine cycle to execute if everything goes well, i.e., no cache misses, no pipeline stalls, no misspeculation, or it may take 100 cycles is everything goes wrong, i.e., cache misses, pipeline stalls, long instruction latencies, long retirement. The variability of execution times exists on all system layers, not only in the processor architecture, but also the software development for single tasks, the task-coordination level and distributed operation [5]. Approaches to improve the average case behavior of systems are often disastrous to predictability.

Actual execution times depend on the execution state of the system, which are determined by the execution history.

The *Timing-Analysis* problem consists in the determination of *safe* and *precise* bounds on the execution times of *all* runs of a system.

Timing Analysis – our Solution

Timing Analysis of real-time systems is a lively research area. A number of commercial tools and academic prototypes have been developed [6]. One of the commercial tools, AbsInt's aiT tool, is in routine use in the areonautics and the automotive industries [4, 3, 2]. It is used in the certification of several time-critical avionics systems.

aiT uses static program analysis to determine strong invariants at each program point about sets of executions reaching this point. These invariants allow the prediction of safe and precise upper bounds of the instruction at this program point.

The control flow of the program is translated into an integer linear program whose maximal solution determines an upper bound on the execution of the whole program and identifies the control-flow path on which this bound was computed. executions

Experience

Valuable experience has been made on the user and on the toolmaker side. Most tools are based on an abstract model of the underlying hardware. These models are costly to build and hard to get correct. Formal methods are currently being developed to derive them in an efficient way guaranteeing correctness by construction.

Experience with industrial use has shown that the difficulty of deriving guarantees strongly depends on the *timing-predictability* properties of the systems, in particular of the employed processor architecture, the software design discipline, the operating system including the scheduling strategy, and the communication mechanism.

Reconciling Predictability with Performance – the Vision

Claim 1 *Over-provisioning will no longer work*

Traditionally, one tries to give guarantees on the worst case or critical case behavior by increasing the average case performance (over-provisioning). However, variability of execution times has become too large to use over-provisioning to derive guarantees.

Claim 2 *Completely deterministic systems will not perform*

A conservative strategy throughout all design decisions attempts to arrive at completely deterministic designs. such as the time-triggered architecture, the damnation of caches, pipelining, speculation and dynamic scheduling. This approach favors predictability but suffers from poor average-case performance by ignoring the advances in computer architecture design.

Rule 1 *Design only what you can analyze*

Predictability needs off-line analysis of whole hardware/software systems. Therefore, all design and implementation methods need to be considered under the aspects of analzability, i.e., analysis complexity and precision [1]. Basic questions concern the relation between analysis and design such as: What is the influence of design decisions (e. g. changing cache replacement from LRU to PLRU) on the variability of execution times? How is the distribution of worst and average case execution times? Which design methods combine well with which analysis methods? How are design decisions affected by the need to provide analyzability?

Rule 2 *Bound the penalties*

Penalties have to be paid for the uncertainty remaining after analysis, e.g. a cache-miss penalty has to be paid for any memory reference that cannot be classified as a cache hit. Some system features introduce huge or even unbounded penalties, e.g. virtual memory, stochastic network protocols, or service brokerage. Careful design must lead to bounded penalties of acceptable magnitude if predictability should result.

Rule 3 *Use resource-aware design*

System design has traditionally profited from principles such as the *separation of concerns* and the *abstraction from resources*. The abstraction from machine time was the most significant. However, these same design principles are conflicting with the design goal Predictability. A new principle for the design of layered systems, *resource-aware abstraction* has to be developed and used for the construction of time-critical systems.

References

[1] R. Heckmann, M. Langenbach, St. Thesing, and R. Wilhelm. The influence of processor architecture on the design and the results of WCET tools. *IEEE Proceedings on Real-Time Systems*, 91(7):1038–1054, 2003.

[2] P. Montag, S. Goerzig, and P. Levi. Challenges of timing verification tools in the automotive domain. In T. Margaria, editor, *2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, 2006.

[3] J. Souyris, E. Le Pavec, G. Himbert, V. Jgu, G. Borios, and R. Heckmann. Computing the worst case execution time of an avionics program by abstract interpretation. In *Proceedings of WCET 2005*, 2006.

[4] St. Thesing, J. Souyris, R. Heckmann, F. Randimbivololona, M. Langenbach, R. Wilhelm, and C. Ferdinand. An abstract interpretation-based timing validation of hard real-time avionics software systems. In *Proceedings of the Performance and Dependability Symposium, San Francisco, CA*, June 2003.

[5] L. Thiele and R. Wilhelm. Design for timing predictability. *Real-Time Systems*, 28:157 – 177, 2004.

[6] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, , and P. Stenström. The worst-case execution time problem - overview of methods and survey of tools. under revision for ACM Transactions on Embedded Computing Systems.