Timing Analysis: In Search of Multiple Paradigms

Frank Mueller
Department of Computer Science/
Center for Embedded Systems Research
North Carolina State University
448 EGRC, Raleigh, NC 27695-7534
e-mail: mueller@cs.ncsu.edu, phone: +1.919.515.7889, fax: +1.919.515.7925

Abstract

While timing analysis is a vital prerequisite for real-time schedulability tests, prior advances on timing analysis were sometimes incremental in addressing selected new processor features or limited source code annotations to provide safe but tight WCET bounds. In contrast, recent advances in timing analysis including a refreshing set of novel approaches to bound the worst-case execution time (WCET) of real-time tasks. These advances are in search of new methodologies to address the timing analysis problem and approach this problem in a more general manner. Examples are 1) the complexity wall of hardware that causes timing analysis tools to trail behind the curve of microarchitectural innovation, 2) severe restrictions on the knowledge of loop bounds, 3) the inability to analysis large application programs, 4) a lack of capitalizing on opportunities of dynamic scheduler interactions and 5) the challenge in expressing certainty levels of WCET bounds.

This paper gives an overview of selected approaches and contributes an initial account on the potential of the contributions for the field. Each of these novel approaches to timing analysis solves one problem in current toolsets. It appears that the diversity of approaches is a valuable asset to the research area. While a particular solution may prove best suitable for some problem, another approach may be required for different problems. Furthermore, many of the recent advances are complementing each other. Some can be used in conjunction to provide additional robustness to hard real-time systems.

1. Introduction

Schedulability analysis of real-time systems determines if given a set of tasks can be feasibly scheduled such that every deadline is met. The specification of task sets includes, at a minimum, the period and the worst-cases execution time (WCET) of each task. While the period is typically derived from the application environment, determining the WCET remains a hard problem, particularly for complex application codes with large inputs and advanced architectures with sources of unpredictable behavior.

The objective of timing analysis is to provide upper bounds on the worst-case execution time (WCET) of hard real-time tasks. By designing timing analysis tools to *safely and tightly* predict the WCET, much of the challenge of timing analysis can be addressed in an automated fashion. Nonetheless, timing analysis has, to date, only been applied to small codes and simple architectures. Manual annotations are often required to supplement source code with constraints necessary to bound selected loop iterations and describe flow dependencies in order to derive tight WCET bounds. Larger programs or programs with complex inputs typically exceed the capability of analysis tools due to non-linear analysis overheads. Advances in microarchitecture create a complexity wall, which causes timing analysis tools to generally trail years behind the curve of architectural innovations.

To address these problems in timing analysis, recent research has taken fresh approaches at the problem. Architectural enhancements have been proposed to counter the complexity wall in processor innovations. Methods of parametric timing analysis have been explored to handle loops with variable bounds and provide late bindings for the actual number of loop iterations. Modular analysis methods have been proposed to address the analysis constraints on large programs. Instead of requiring tight WCET bounds, looser bounds are deemed sufficient for dynamic voltage scaling (DVS) scheduling algorithms that trade speed for power conservation. Frequency-aware static timing analysis (FAST) complements DVS scheduling for real-time systems by ensuring the timing predictions do not deteriorate to a level where power consumption is sacrificed because of inaccuracies in bounding the WCET. While static timing analysis approaches can be complemented by experimental timing values, *e.g.*, through genetic algorithms to search for longer execution times as a function of the input, recent work is promoting extreme statistics in conjunction with observed timing measurements in a probabilistic WCET assessment. Resulting WCET bounds are guaranteed with a certainty of some probability.

In the following, an overview of some of these novel approaches is given. Some of these research advances solve one problem in current timing analysis toolsets. The diversity of approaches provides a valuable asset to the research area in large. While a particular solution may prove best suitable for some problem, another approach may be required for different problems. Furthermore, many of the recent advances are complementing each other. For example, DVS scheduling and FAST can be used in conjunction to provide additional robustness to hard real-time systems.

2. VISA: A Virtual Simple Architecture

VISA resolves a long-standing problem in embedded systems: bounding the worst-case execution times (WCET) of tasks on contemporary processors. WCETs are essential for real-time scheduling; yet deriving them for contemporary processors is intractable. The Virtual Simple Architecture (VISA) framework shifts the burden of bounding the WCETs of tasks, in part, to hardware. A VISA is the pipeline timing specification of a hypothetical simple processor. WCET is derived for a task assuming the VISA. At run-time, the task is executed speculatively on an unsafe complex processor, and its progress is continuously gauged. If continued safe progress appears to be in jeopardy, the complex processor is reconfigured to a simple mode of operation that directly implements the VISA, thereby explicitly bounding the task's overall execution time by the WCET.

System design involves defining a VISA, and then implementing a static timing analyzer and a complex processor, the latter two compliant with the VISA. In defining the VISA, we considered the capabilities of current timing analysis tools. We also considered how a simple mode of operation is likely to be accommodated within a typical dynamically scheduled superscalar processor and, in particular, the complex processor described in [1]. The three layers — VISA, processor, and timing analyzer — are shown in Fig. 1 and are described in detail elsewhere [1].

Typically, the VISA-protected complex processor finishes periodic hard-real-time tasks faster than an explicitly-safe simple processor, creating slack in the schedule. Dynamic slack cannot be used to schedule more periodic hard-real-time tasks, because WCETs are based on the hypothetical simple processor. Nonetheless, slack can be exploited to safely lower frequency/voltage for power savings. This application was explored in our recent ISCA'03 paper [1], which demonstrated 43-61% power savings.

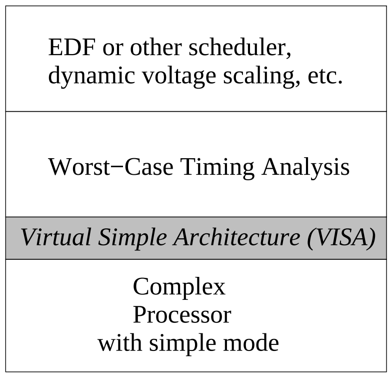
EDF or other scheduler,
dynamic voltage scaling, etc.
Worst−Case Timing Analysis
Virtual Simple Architecture (VISA)
Complex
Processor
with simple mode

Figure 1. VISA Abstraction Layers

In summary, VISA provides a general framework for safe operation on unsafe processors, setting up new opportunities for exploiting higher performance in embedded systems.

3. Parametric Timing Analysis

Traditionally, static timing analysis yields a discrete prediction of the WCET based on fixed iteration bounds that have to be known statically. Such analysis requires that the actual number of iterations always be known prior to execution of a real-time task.

This work addresses these limitations of timing analysis for embedded systems. It contributes a novel approach to program analysis through parametric techniques of static timing analysis and provides innovative methods for exploiting them. The proposed techniques express worst-case execution time as a formula with the following benefits. First, static timing analysis becomes applicable to a wide class of embedded systems with variable loop bounds. Second, the techniques of static timing analysis, currently constrained to hard real-time system, become applicable to a much wider range of embedded systems with soft timing constraints.

For example, a loop may encompass n iterations, where n depends on the current input. Typically, the number of iterations n is already known prior to entering the loop. Static timing analysis may supply a closed formula to predict the WCET for the remaining execution time based on dynamic information, such as the actual value of n.

A novel fix-point algorithm has been developed to derive closed formulas for expressing the WCET bounds on loops with variable trip counts. Preliminary experimental results using parametric analysis in contrast to the traditional numerical technique illustrate the feasibility this novel approach [7]. Parametric timing analysis only looses an insignificant amount of precision (tightness) in bounding the

WCET. The benefit of flexibility outweights this diminishing cost. Applications of the approach lie in novel applications for dynamic scheduling decisions and a potential for more effective system utilization.

4. Modular Static Cache Simulation

Static timing analysis techniques derive safe WCET bounds via path analysis, pipeline simulation and cache simulation. But such analysis has traditionally been constrained to only small programs due to the complexity of simulation, most notably by static cache simulation, which requires inter-procedural analysis. A novel approach of modular static cache simulation is devised that alleviates the complexity problem, thereby making static timing analysis feasible for much larger programs that in the past.

The main contribution of this effort is a framework to perform static cache analysis in two steps, a module-level analysis and a compositional phase, thus addressing the issue of complexity of inter-procedural analysis for an entire program. The module-level analysis parameterizes the data-flow information in terms of the starting offset of a module. The compositional analysis stage uses this parameterized data-flow information for each module. Thus, the emphasis here is on handling most of the complexity in the module-level analysis and performing as little analysis as possible at the compositional level. The experimental results for direct-mapped instruction caches show that the compositional analysis framework provides equally accurate predictions when compared with the simulation approach that uses complete inter-procedural analysis [5]. This novel approach to static cache analysis provides a promising solution to the complexity problem in timing analysis, which, for the first time, makes the analysis of larger programs feasible.

5. DVS: Dynamic Voltage Scheduling

Dynamic voltage scaling (DVS) is a promising method for embedded systems to exploit multiple voltage and frequency levels and to prolong battery life. Numerous DVS scheduling algorithms for real-time systems have been proposed, but the most promising approaches are those that capitalize slack reclamation due to idle time (static slack) and, in addition, slack due to early completion (dynamic slack). This requires a dynamic scheduling scheme, such as EDF enhanced by slack reclamation and voltage/frequency modulation, *e.g.*, Look-ahead EDF [6]. The benefit for DVS scheduling for real-time systems is that even loose WCET bounds suffice since voltage/frequency modulation is based on the actually observed execution times in dynamic scheduling schemes. Notice that static DVS schemes cannot be as aggressive and, hence, should receive less attention. Furthermore, DVS scheduling can, in general, profit from deviations between WCET and actual execution times, which are reportedly significant with the actual execution time of real-time tasks ranging from about 30% to 90% of the WCET, depending on the application.

Pure DVS techniques do not perform well for dynamic systems where the system workloads vary significantly. A novel approach combining feedback control with DVS schemes was devised targeting hard real-time systems of dynamic workloads. This method relies strictly on operating system support by integrating a DVS scheduler and a feedback controller within the EDF scheduling algorithm. The scheme splits each task into two portions, and aims at finishing the actual execution within the first portion of each task while reserving enough time in the second portion to meet the deadline following a last-chance approach. Feedback techniques make the system capable to select the right frequency and voltage settings for the first potion, as well as guaranteeing the deadline requirements of the hard real-time task sets. Simulation experiments demonstrate that our algorithm is able to saves up to 29% more energy than previous work for task sets with different dynamic workload characteristics [10, 3, 9].

6. FAST: Frequency-Aware Static Timing Analysis

DVS techniques for real-time scheduling focus on saving power in static as well as dynamic scheduling environments by exploiting idle and slack due to early task completion for DVS of subsequent tasks. As mentioned before, these approaches rely on *a priori* knowledge of the WCET for each task. All DVS scheduling algorithms assume that DVS has no effect on the worst-case execution cycles (WCEC) of a task and scale the WCET according to the processor frequency. However, for systems with memory hierarchies, the WCEC typically *does* change under DVS due to frequency modulation. Hence, current assumptions used by DVS schemes result in a highly exaggerated WCET.

A novel technique for tight and flexible static timing analysis is developed that is particularly well-suited for dynamic scheduling schemes. The technical contributions are as follows: (1) The problem of changing execution cycles due to scaling techniques is assessed. (2) A parametric approach towards bounding the WCET statically with respect to the frequency is proposed. Using a parametric model, one can capture the effect of changes in frequency on the WCEC and, thus, accurately model the WCET over any frequency range. (3) Design and implementation of the frequency-aware static timing analysis (FAST) tool are discussed based on prior experience with static timing analysis. (4) Experiments demonstrate that the FAST tool provides safe upper bounds on the WCET, which are tight.

The tool provides the means to capture the WCET of six benchmarks using equations that overestimate the WCET by less than 1%. FAST equations can also be used to improve existing DVS scheduling schemes to ensure that the effect of frequency scaling on WCET is considered and that the WCET used is *not* exaggerated. (5) Three DVS scheduling schemes are leveraged by incorporating FAST into them and by showing that the power consumption further decreases. Overall, FAST and DVS scheduling techniques for real-time systems provide a symbiotic solution to aggressive lower power consumption without missing deadlines – and to cope with (as well as benefit from) looser WCET bounds.

7. Probabilistic Timing Analysis

Traditional timing analysis techniques rely on static analysis of programs. Past work has discussed the benefits of complementing static and dynamic analyses for soft and hard real-time systems, most notably by guiding the search for the WCET in dynamic analysis via genetic algorithms [4, 8]. Taking dynamic analysis one step further, recent work promotes a probabilistic approach to provide certainty levels for the WCET. These certainty levels, expressed as probabilities, are based on a sample of observed executions and combined with extreme-value statistics. The supporting experimentation tool, the pWCET tool, handles timing at a basic block level and combines analysis with the traditional timing schema [2]. Such an approach can also be combined with the VISA architecture to obtain WCET bounds for the simple mode. VISA then provides safety by gaging progress in complex mode since non of the above static analysis techniques can fully capture the effects of a complex architecture. Even the pWCET results at some certainty level can profit from the additional safety provided by VISA.

8. Conclusion

This paper provided an overview of a variety of selected novel approaches in tackling the problem of providing bounds for the WCET of real-time tasks. These research advances each solve some problem in current timing analysis toolsets. The diversity of approaches provides a valuable asset to the research area in large. While a particular solution may prove best suitable for some problem, another approach may be required for different problems. Furthermore, many of the recent advances are complementing each other. For example, DVS scheduling and FAST as well as VISA and pWCET can be used in conjunction to provide additional robustness to hard real-time systems. Overall, the field of timing analysis has matured and, at the same time, is still open to new discoveries with potentially significant impact.

References

[1] A. Anantaraman, K. Seth, K. Patil, E. Rotenberg, and F. Mueller. Virtual simple architecture (VISA): Exceeding the complexity limit in safe real-time systems. In *International Symposium on Computer Architecture*, pages 250–261, June 2003.

[2] G. Bernat, A. Collin, and S. Petters. Wcet analysis of probabilistic hard real-time systems. In *IEEE Real-Time Systems Symposium*, Dec. 2003.

[3] A. Dudani, F. Mueller, and Y. Zhu. Energy-conserving feedback edf scheduling for embedded systems with real-time constraints. In *ACM SIGPLAN Joint Conference Languages, Compilers, and Tools for Embedded Systems (LCTES'02) and Software and Compilers for Embedded Systems (SCOPES'02)*, pages 213–222, June 2002.

[4] F. Mueller and J. Wegener. A comparison of static analysis and evolutionary testing for the verifi cation of timing constraints. In *IEEE Real-Time Technology and Applications Symposium*, pages 179–188, June 1998.

[5] K. Patil. Compositional static cache analysis using module-level abstraction. Master's thesis, Dept. of CS, North Carolina State University, Aug. 2003.

[6] P. Pillai and K. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Symposium on Operating Systems Principles*, 2001.

[7] E. Vivancos, C. Healy, F. Mueller, and D. Whalley. Parametric timing analysis. In *ACM SIGPLAN Workshop on Language, Compiler, and Tool Support for Embedded Systems*, volume 36 of *ACM SIGPLAN Notices*, pages 88–93, Aug. 2001.

[8] J. Wegener and F. Mueller. A comparison of static analysis and evolutionary testing for the verifi cation of timing constraints. *Real-Time Systems*, 21(3):241–268, Nov. 2001.

[9] Y. Zhu and F. Mueller. Preemption handling and scalability of feedback dvs-edf. In *Workshop on Compilers and Operating Systems for Low Power*, Sept. 2002.

[10] Y. Zhu and F. Mueller. Feedback dynamic voltage scaling dvs-edf scheduling: Correctness and pid-feedback. Technical Report TR 2003-13, Dept. of Computer Science, North Carolina State University, 2003.