03471 Abstracts Collection
Design of Systems with Predictable Behaviour
— Perspectives Workshop —

Lothar Thiele[1] and Reinhard Wilhelm[2]

[1] ETH Zürich, Institut für Techn. Informatik und Kommunikationsnetze
8092 Zürich, Gloriastr. 35, Switzerland
thiele@tik.ee.ethz.ch
[2] Universität des Saarlandes, FR Informatik
66041 Saarbrücken, PF 15 11 50, Germany
wilhelm@cs.uni-sb.de

On 16.11.-19.11.2003, the Perspectives Workshop 03471 "Design of Systems with Predictable Behaviour" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl. During the workshop, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well a digest of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords: Real-time systems; guarantees; predictability; embedded systems; performance

Design for Time-Predictability

Reinhard Wilhelm (Universität Saarbrücken)

A large part of safety-critical embedded systems has to satisfy hard real-time constraints. These need sound methods and tools to derive reliable run-time guarantees. The guaranteed run times should not only be reliable, but also precise. The achievable precision highly depends on characteristics of the target architecture and the implementation methods and system layers of the software. Trends in hardware and software design run contrary to predictability. This article describes threats to time-predictability of systems and proposes design principles that support time predictability. The ultimate goal is to design performant systems with sharp upper and lower bounds on execution times.

Keywords: Real-time systems; guarantees; predictability; embedded systems; performance

Joined work with: Lothar Thiele (ETH Zürich)

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2004/2/

Requirements for and Design of a Processor with Predictable Timing

Christoph Berg (Universität Saarbrücken)

This paper introduces a set of design principles that aim to make processor architectures amenable to static timing analysis. Based on these principles, we give a design of a hard real-time processor with predictable timing, which is simultaneously capable of reaching respectable performance levels.

The design principles we identify are recoverability from information loss in the analysis, minimal variation of the instruction timing, non-interference between processor components, deterministic processor behavior, and comprehensive documentation. The principles are based on our experience and that of other researchers in building timing analysis tools for existing processors.

Keywords: WCET, hard real-time, embedded systems, computer architecture

Joined work with: Engblom, Jakob; Wilhelm, Reinhard

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2004/5/

Strict Isolation of Real-Time Threads on Multithreaded Hardware

Uwe Brinkschulte (Universität Karlsruhe)

Highly dynamic programming environments for embedded real-time systems require a strict isolation of real-time threads from each other to achieve dependable and predictable systems. We propose a new real-time scheduling technique, called guaranteed percentage (GP) scheme that assigns each thread a specific percentage of the processor power. A hardware scheduler in conjunction with a multithreaded processor guarantees the execution of instructions of each thread according to their assigned percentages within a time interval of 100 processor cycles.

We compare performance and implementation overhead of GP scheduling against fixed priority preemptive (FPP), earliest deadline first (EDF), and least laxity first (LLF) scheduling using several benchmarks on our Komodo microcontroller that features a multithreaded Java processor kernel. Our evaluations show that GP scheduling reaches a speed-up similar to EDF and FPP but worse than LLF. However, its hardware implementation costs are still reasonable, whereas the LLF overhead is prohibitive. Only GP reaches the isolation goal among the examined scheduling schemes.

Furthermore, we propose a control theory based approach to compensate latencies. This increases the predictability of the execution of a single GP thread. The performance of the thread in terms of true active cycles (AC) or executed instructions per cycle (IPC) is monitored and the guaranteed percentage rate of this thread is adapted in a closed control loop to reach the requested AC or IPC value.

Finally, we present how guaranteed percentage scheduling can be used to reduce the energy consumption of the microcontroller. If the requested percentage of all threads on the microcontroller is below 100%, the clock frequency and the supply voltage can be automatically reduced. This results in a cubic energy saving while the runtime behavior will not change.

Joined work with: Ungerer, Theo

Long Timing Effects

Jakob Engblom (Uppsala University)

This talk discusses the occurrence of Long Timing Effects (LTE) in processor pipelines. Basically, an LTE occure when the timing of a particular instruction depends on a non-adjacent instruction in the program. The more LTEs occur in a program, the harder it will be to analyze, as the analysis effort has to include more of the program. The occurence of LTEs is caused by the particular structure of the pipeline and latencies of instructions in a processor. It is a property of the processor, even if cleverly written programs can avoid most LTEs.

Simplicity Considered Fundamental to Design for Predictability

Wolfgang A. Halang (FernUniversität Hagen)

Complexity is the core problem of contemporary information technology, as the "artificial complicatedness" of its artefacts is exploding. Intellectually easy and economically feasible predictability can be achieved by selecting implicity as fundamental design principle. Predictability of system behaviour is identified as the central concept for the design of real-time and embedded systems, since it essentially implies the other requirements timeliness and ependability holding for them. Practically all dynamic and "virtual" features aiming to enhance the average performance of computing systems as well as the traditional categories and optimality criteria are found inadequate and are, thus, considered harmful. In mainstream research on scheduling the gap between academic research and reality has grown so wide that research results are doomed to irrelevance. Instead, useful scheduling research ought to employ utmost simplicity as optimality criterion, and strive to minimise software size and complexity. Computing should

embrace other disciplines' notions and technologies of time. Programming and verification methods for safety-related applications are identified on the basis of their simplicity and ergonomic aptitude. It is advocated to utilise the permanent advances in microelectronics to solve long untackled problems and to foster simplicity and predictability by hardware support.

Keywords: Design for predictability, simplicity, dependability, safety, real-time and embedded systems, design concepts

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2004/4/

The Influence of Processor Architecture on Worst-Case Execution Time Analysis

Reinhold Heckmann (AbsInt - Saarbrücken)

After a brief presentation of AbsInt, we concentrate on cache analysis and demonstrate the influence of the cache replacement strategy on the predictability of the cache contents. Best results are achieved with LRU caches where a sequence of known accesses leads to complete knowledge of the cache contents even when starting from a completely unknown cache. In contrast, we can predict at most half of the contents of the Pseudo LRU caches of the PowerPC 750/755, and at most one quarter of the Pseudo Round Robin cache of the ColdFire MCF 5307. Finally, we present a list of other features that positively or negatively influence WCET predictability.

Keywords: Worst-case execution time, cache analysis

Compiler-Support for scratchpad Memories makes Memory Access Times predictable

Peter Marwedel (Universität Dortmund)

The design of future high-performance embedded systems is hampered by two problems: First, the required hardware needs more energy than is available from batteries. Second, current cache-based approaches for bridging the increasing speed gap between processors and memories cannot guarantee predictable real-time behavior. A contribution to solving both problems is made in this paper which describes a comprehensive set of algorithms that can be applied at design time in order to maximally exploit scratch pad memories (SPMs). We show that both the energy consumption as well as the computed worst case execution time (WCET) can be reduced by up to to 80% and 48%, respectively, by establishing a strong link between the memory architecture and the compiler.

Keywords: Embedded system, compiler, energy efficiency, low power, WCET, scratchpad, memory access

Joined work with: Wehmeyer, L.; Verma, M.; Steinke, S; Helmig, U.

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2004/6/

A Multithreaded RISC/DSP Processor from the Worst Case Execution Point of View

Erik Norden (Infineon Technologies - München)

To satisfy the need for high-performance in low-cost embedded applications, the right mix of processor and system features as well as an innovative design is crucial.

Infineon has developed a new processor solution based on its TriCore Unified Processor architecture: TriCore 2. Advanced pipeline technology allows high instruction per cycle (IPC) performance while reaching higher frequencies and complying with demanding automotive requirements. Multithreading enables the full utilization of the execution pipelines during code fetch latencies and helps to save costs and power consumption. The center of the processor's hierarchical memory subsystem is an open, scalable crossbar architecture, which provides a method for efficient parallel communication to code and data memory including multiprocessor capability.

For mission critical real-time applications, the worst case execution behavior is a key factor. Different features of the TriCore Architecture reduce the penalty and simplify the predictability like: scratch memory, interrupt system, pipeline features and more.

Goal-Oriented Design for Predictable Timing

Peter Puschner (TU Wien)

We present an extreme approach towards achieving temporal predictability in real-time systems: In order to be predictable we propose that a computer system exercise full control over all its actions and the timing of its actions instead of being controlled by the environment (e.g., by input data or interrupt signals). Computer systems for which task activation, communication, and synchronization only happen at points in time that have been exactly planned during system design form the basis of this approach. We extend this by a task execution model in which the actions to be executed and the timing of these actions must not be dictated by the environment, either. This is achieved by converting all tasks to so-called single-path code that has invariable execution time. Only these single-path tasks are allowed to execute on the system. The knowledge about the exact behaviour of such a fully predictable real-time computer system makes it possible to use performance-enhancing features that do not destroy the predictability of the system. Thus we end up with systems that are both fully predictable wrt. timing and well performing.

Towards predictable high-performance processors

Christine Rochange (IRIT - Toulouse)

In his presentation, Jakob Engblom has shown that high-performance pipelines are generally not compatible with safe computation of the Worst-Case Execution Time because of the possible long timing effects.

Im my talk, I show how those advanced pipelines could be made predictable and that we do not have to resign ourselves to low-performance architectures.

The main idea is that each piece of code that we would like to measure as part of WCET estimation (typically a basic block) should execute as if it was alone in the pipeline. For that purpose, we include in the processor hardware a module called fetch controllerthta observes the pipeline and decides when a new block can safely enter the pipeline (i.e. it cannot be stalled by nor itself stall another basic block). This module needs to maintain tables that store the availability dates of any kind of resource (pipeline stage, functional unit, register to be used as an operand, ...), and the condition for fetching a new basic block is those dates match the potential requirements of any new instruction.

The feasibility of maintaining such availability tables relies on the instruction scheduling policy within the pipeline. With in-order scheduling, they can be updated as the instructions enter the issue buffer, while out-of-order scheduling would require to recompute all availabilities at each cycle because any instruction can change the scheduling of previous ones. This would be very complex and could probably be implemented. An another scheduling strategy, called dynamic prescheduling has been proposed to reduce the complexity of the scheduling logic (Canal and Gonzalez, Intl Conf. on Supercomputing, 2000). It completely suits our scheme since it sets the instruction scheduling order at issue time and then allows the computation of the resource availabilities at the same time.

First simulation results show that the joint cost of dynamic prescheduling and fetch controlling is a less to 15% mean speed-down compared to pure out-of-order scheduling without fetch control.

Thoughts on Modular Performance Estimation

Lothar Thiele (ETH Zürich)

The talk introduced a unified approach to the performance analysis of distributed systems. Major emphasis was (1) to cover computation and communication, (2) to consider hard real-time bounds, (3) to take into account the run-time system and (4) to present a modular approach. The latter aspect was considered by proposing a formal method based on the concept of components and interfaces. Finally, to a component consisting of hardware, software and an operating system one can associate an interface specification that reflects the properties of resource usage, load and delay.

In a second part, an instance of this new method was presented, based on the concpets of service curves, arrival curves, and a new real-time calculus. Finally, it was shwon that the obtained worst case bounds in terms of delay, memory and throughput are close to simulation results.

Strict Isolation of Real-Time Threads on Multithreaded Hardware

Theo Ungerer (Universität Augsburg)

Highly dynamic programming environments for embedded real-time systems require a strict isolation of real-time threads from each other to achieve dependable and predictable systems. We propose a new real-time scheduling technique, called guaranteed percentage (GP) scheme that assigns each thread a specific percentage of the processor power. A hardware scheduler in conjunction with a multithreaded processor guarantees the execution of instructions of each thread according to their assigned percentages within a time interval of 100 processor cycles.

We compare performance and implementation overhead of GP scheduling against fixed priority preemptive (FPP), earliest deadline first (EDF), and least laxity first (LLF) scheduling using several benchmarks on our Komodo microcontroller that features a multithreaded Java processor kernel. Our evaluations show that GP scheduling reaches a speed-up similar to EDF and FPP but worse than LLF. However, its hardware implementation costs are still reasonable, whereas the LLF overhead is prohibitive. Only GP reaches the isolation goal among the examined scheduling schemes.

Furthermore, we propose a control theory based approach to compensate latencies. This increases the predictability of the execution of a single GP thread. The performance of the thread in terms of true active cycles (AC) or executed instructions per cycle (IPC) is monitored and the guaranteed percentage rate of this thread is adapted in a closed control loop to reach the requested AC or IPC value.

Finally, we present how guaranteed percentage scheduling can be used to reduce the energy consumption of the microcontroller. If the requested percentage of all threads on the microcontroller is below 100%, the clock frequency and the supply voltage can be automatically reduced. This results in a cubic energy saving while the runtime behavior will not change.

Joined work with: Brinkschulte, U.

The TIMES tool: Schedulability Analysis and Code Synthesis (www.timestool.com)

Wang Yi (University of Uppsala)

We present the design and theoretical foundations for TIMES, a software tool for schedulability analysis and automatic synthesis of real time software guaranteeing timing constraints. In classic scheduling theory, real time tasks (processes) are usually assumed to be periodic, i.e. tasks arrive (and will be computed) with fixed rates periodically. Analysis based on such a model of computation often gields pessimistic results. To relax the stringent constraints on task arrival times, we propose to use automata with timing constraints to model task arrival patterns. We show that the general schedulability checking problem for such models is decidable. In this talk, we show that for fixed priority scheduling strategy, the problem can be efficiently solved by reachability analysis on timed automata using only 2 extra clock variables. The analysis can be done in a similar manner to response time analysis in classic Rate-Monotonic Scheduling. TIMES is a tool developed based on these recent results and our past experience in developing UPPAAL, a model checker for real time systems.