Submission to
The ASPLOS Wild and Crazy Idea Session
6th October 1998

Title: **A Proposal for a Real-Time Processor Core**

Authors: Claudia Mathis, Reinhold Weiss

Affiliation: Institute for Technical Informatics
Technical University of Graz
Steyrergasse 17/4
A-8010 Graz / Austria

Corresponding author:
Claudia Mathis
Tel: +43 316 873 6404
Fax: +43 316 873 6903
e-mail: mathis@iti.tu-graz.ac.at

Keywords: real-time processor core, hardware assisted scheduling, real-time

Abstract:

The problem is:

Predictability is fundamental for real-time systems. Without predictability, tasks with hard deadlines cannot be guaranteed. But advanced architectures usually comprise components, like caches, or employ pipelining, which provides high performance on the average. Unfortunately, the stochastic nature of these architectures makes it nearly impossible to exactly determine the execution times of tasks or their run-time behavior, respectively. The modeling of the underlying hardware architecture and its behavior in order to ascertain the worst case task execution times for predictability reasons, is complex and difficult.

Further, tasks have to be scheduled and their deadlines be guaranteed by the operating system. This task management is necessary, but consumes CPU-time, which is of value to the real-time tasks and, therefore, should be kept to a minimum. In real-time systems task execution is usually based on periodic timer interrupts. Upon occurrence of a timer interrupt a call to the timer interrupt service routine is performed. Interrupt latency also has to be taken into account. Then a scheduler has to decide which task is executed next. And, finally, if a different task is to be run, a context switch will be carried out. Performing these system tasks causes system overhead which is not desirable. The modeling of task execution in view of schedulability, will also be more complex and difficult, if system overhead is included. Nevertheless, this sophisticated modeling will be necessary, if real-time scheduling theory is applied to real-time system implementation.

The solution is:

Thus, the idea is to have a simple and small, but fast and predictable processor core. It is, therefore, reduced in functionality and consists of only a few very basic elements needed. It does not comprise components, like caches, or makes use of pipelining, which both may cause a variance in task execution times.

It further supports fast context switching and an appropriate memory interface to a designated context RAM all the task contexts are stored in. Both, the context switching and the memory interface, are controlled by a dispatcher implemented in hardware.

The most important part added to the processor core is a scheduler, which is also implemented in hardware. It contains the task description information of each task and obeys to a real time scheduling algorithm, e.g. the rate monotonic scheduling algorithm.
The major drawback of this processor core would be, that only a limited number of tasks can be executed. This is due to the hardware implementation of the scheduler and depends mainly on the fact, that the higher the number of tasks executed and, therefore, scheduled, the higher the hardware costs are. This can be opposed by the following: Real-time control applications in general consist of a limited and small set of tasks. And, very high integration densities will be available in future allowing larger sets of tasks.

The justification and conclusions are:
Predictability, which is important for real-time systems, is established by the simple processor core. Consequently, the core is easy to model, task execution times and behavior can be determined exactly.
System overhead is completely removed because of hardware implemented scheduling and context switching. This also allows the use of simpler models for schedulability analyzability.
Timer interrupts are superfluous, as requests for tasks and their timeliness are managed by the scheduler.
Interrupt service routines are equal to normal tasks. They have their own context, timing requirements, and, priority to range them under the ready tasks which are to be executed.
Implementation and maintenance of real-time systems based on this processor core would be much easier because of the processor concept and the provided real-time scheduling algorithm.
Finally stated, quantitative analysis is trivial. (How to show that a system overhead of zero is best?)