- From Single to Multiprocessor Real-Time Kernels in Hardware -

Lennart Lindh, Johan Stärner and Johan Furunäs
Mälardalens University
IDt/ Department of Real-Time Computer Systems
P O Box 883, S-721 23 Västerås, Sweden
Fax: xx46 21 101460
email: lennart.lindh@mdh.se, johan.starner@mdh.se, johan.furunas@mdh.se

Abstract

This article presents three different implementations of a traditional Real-Time Kernel in hardware. All approaches improved performance and determinism by several orders of magnitude when compared with software-based real time kernels.

The first implementation provides an integrated deterministic CPU and a deterministic and high performance multitasking real time kernel in hardware. The second implementation provides a deterministic and high performance standalone multitasking real time kernel in hardware and the last implementation provides a deterministic and high performance real time kernel for Homogeneous and Heterogeneous Multiprocessor Real-Time Systems.

1. Introduction

The real time kernel is most commonly implemented in machine code which resides in the primary memory. Considerable predictability problems are encountered in these systems because of the non-deterministic behaviour of the kernel. μ-coded real time kernels are implemented directly in the CPU. The implementation is similar to that of machine code except that the operations of the real time kernel are performed more rapidly. One advantage is that the micro program has direct access to the data paths and the working units (such as ALU) in the CPU.

The separation of the real time kernel from the CPU raises the performance of and improves the predictability in Real-Time systems. The processor load is reduced and new possibilities are created for implementing real time functions without affecting the performance of the CPU.

The implementation of a real time kernel in a state machine (hardware) is an acceleration of its implementation in a von Neuman architecture CPU. A coprocessor for multi-processor scheduling in the SPRING project is implemented in a state machine with data paths [Spring93]. The state machine controls several "working units" simultaneously. The hardware architecture is scalable for different numbers of tasks and resources. With an internal clock rate of 100 MHz, a speed increase of two or three orders of magnitude is expected for scheduling tasks, as compared with a separate CPU solution.

2. FASTCHART

This section provides a survey of the integrated Real Time Unit (RTU) and CPU we designate FASTCHART (a **FAS**t **T**ime Deterministic **C**PU and **HA**rdware based **R**eal-**T**ime-kernel).

With FASTCHART [FASTCHART91] we obtain a predictable real time hardware architecture, in which FASTCHART is deterministic with respect to:

- Execution of **CPU** instructions,
- Execution of the real time operating system service.

The FASTCHART has two concurrent running parts. One is a CPU designed for our purposes and the other is the Real Time Unit .

The performance of real time systems is increased by implementing a task switch mechanism in hardware. This has two benefits. Firstly, a task switch can be manage (exchange the register contents) in zero time - this means that one can switch between two tasks without the loss of the CPU time needed to exchange the contents of the CPU registers. Secondly, the task switch becomes deterministic since there is no expenditure of execution time.

The RTU contains a scheduler with a priority scheduling algorithm, a dispatcher which controls the task switch mechanism, two wait queues, one for inactive tasks and one for tasks waiting for a time event and a ready queue for tasks waiting to be executed by the CPU.

1080-1812/95 $04.00 © 1995 IEEE

3. FASTHARD

FASTHARD (a **FAS**t **T**ime deterministic **HARD**ware based real-time kernel) is a standalone Real-Time Kernel which can work with almost all CPU's on the market [FASTHARD93].
The FASTHARD prototype can handle 64 tasks and 8 priority levels.

The following real time functions are implemented in FASTHARD: Interrupt handler, periodic start of tasks, relative delay of tasks, scheduler (priority algorithm), activate and terminate task, on/off task switch and VME bus interface [VME].

4. RTU94

RTU94 is based on the FASTHARD concept [RTU94] . The main new feature of RTU94 is its ability to control multiple heterogeneous CPUs. The number of function calls is increased to include semaphores, event flags and watch dogs. All function calls implemented in FASTHARD have been rewritten to support a multiprocessor environment.

A task can be initialized to execute on a fixed CPU (local tasks), but it is also possible to allow the task to execute on any CPU (global tasks). A local task has all the program information in the local RAM, but the global tasks must have their data region in the global memory.

The scheduling concept is a priority-based, pre-emptive algorithm. The RTU94 consists of three schedulers, one for each CPU. There is one ready queue for each CPU (local queues), and one queue for the tasks that can execute on any CPU (global queue). Each scheduler checks both the local and the global queues. Since the RTU94 has information about the load on each CPU it is possible to have dynamic balance of the task load in the system.

5. Conclusion

A "true" definition of the advantages of a hardware based Real-Time kernel can not be generalised for all aspects, because the requirements and constraints are very different for different applications. Given the qualification concerning applications, we can however observe that utilization of a separate hardware-based real time kernel yields the following advantages:

PERFORMANCE and DETERMINISM
• deterministic multitasking and multiprocessor real time kernel,
• the speed with which real time service calls are executed is increased by several orders of magnitude compared with a conventional CPU-based solution.
• no clock tick interrupt to the CPU is needed.

PORTABILITY
• no memory requirements,
• no special instruction format,
• no special CPU, only one interrupt and an external data bus (FASTHARD and RTU94).

SOFTWARE DESIGN
• simpler software system since the program code for the real time kernel does not occupy CPU memory,
• simpler timing analysis of the system,
• improved understandability when the system is divided into parts (complexity reduction),
• all RTK service calls are isolated in the RTU (no problem with interference between different service call codes),
• no external interrupt handler in the software,
• simplified debugging of the software tasks, since different protection modes are not required,

In principle, the motivation for using an RTU is the same as that for using an arithmetical processor in a computer system, i.e. to improve the performance of the application. An arithmetical processor is specialized in performing fast calculations. An RTU is specialized in quickly performing the calls of a real time operating system.

References

[FASTCHART91] L. Lindh, & F. Stanischewski,
FASTCHART - Idea and Implementation, IEEE press, International Conference on Computer Design (ICCD) Cambridge MIT, USA, 14-16 Oct. 1991.

[FASTHARD93] L. Lindh,
FASTHARD Prototype - A Real-Time-Kernel Implemented In One Chip. IEEE press, Real-Time Workshop, Oulu, 3-5 June, 1993.

[RTU94] RTU Data Book,
RF RealFast AB, Dragverksg 138, S-724 74 Västerås, Sweden, Fax xx46 (0)21 811198

[Spring93] Wayne Burleson, Jason Ko, Douglas Niehaus, Krithi Ramamritham, John A. Stankovic, Gary Wallace and Charles Weems,
The Spring Scheduling Co-processor: A Scheduling Accelerator, In To Appear:Proceedings of the International Conference on ComputerDesign. IEEE, October 1993.

[VME] The VMEbus Specification
ANSI/IEEE STD1014-1987, IEC821 and 297, VMEbus International Trade Association, 10229 N. Scottsdale Road, Suite E Scottsdale, AZ 85253 USA