DEDICATED INSTRUCTIONS TO SUPPORT MULTIPROCESSING ON A EMBEDDED JAVA ARCHITECTURE

L.S.Rosa Jr., A.C.Beck Fo., F.R.Wagner, L.Carro, A.S.Carissimi, A.I.Reis

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
PO Box 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{leomarjr, caco, flavio, carro, asc, andreis}@inf.ufrgs.br

ABSTRACT

This work presents a new instruction set to support the design of schedulers for an embedded Java architecture with reduced area cost. Obtained results for a Round-Robin scheduler, which uses these new instructions to execute the context switch, are presented. Experiments show the impact of schedulers using the instruction set and on the embedded system performance.

1. INTRODUCTION

Embedded operating systems became a key element used in many systems essential to modern life. They are found in all kinds of devices and systems, from high-end routers and switches that keep networks running, to medical devices that keep patients alive, as well as copiers, TV remote controls, factory-automation systems, and even talking dolls [1].

Almost every new system that performs different tasks automatically has an embedded operating system orchestrating the performance of its components [2]. The processor use is managed by a routine of the operating system called scheduler. A scheduler is needed when a single processor must handle different tasks [3].

The goal of this paper is to present a multiprocessing alternative for an architecture that cannot execute two or more processes simultaneously. Thus, new instructions to support context switching were developed, leading to good results.

This paper is organized as follows. Section 2 details the processor architecture, while the implemented instructions are discussed in section 3. The evaluation and conclusions are presented in section 4 and 5, respectively.

2. THE EMBEDDED ARCHITECTURE

Efficient execution of Java programs, especially in embedded systems, can be done by direct execution of Java bytecodes in hardware [4]. This is the main characteristic and intention of the FemtoJava microcontroller.

This processor was designed to application in embedded systems with low power characteristics, and its synthesis is targeted for FPGA devices [5]. Other characteristics of this architecture are: reduced area, reduced instruction count and the capacity to add or remove instructions in its VHDL code.

The FemtoJava architecture is a stack-based machine, and therefore, it does not support multiprocesses. Figure 1 presents the FemtoJava micro architecture

3. INSTRUCTIONS FOR CONTEXT SWITCHING

The scheduler is a basic element for the embedded operating system. If an equipment needs to simultaneously execute more than an application on the same CPU, then the scheduler becomes a part of extreme value for the computer system. Thus, we can implement the virtual CPU concept, where a CPU manages the execution of some processes.

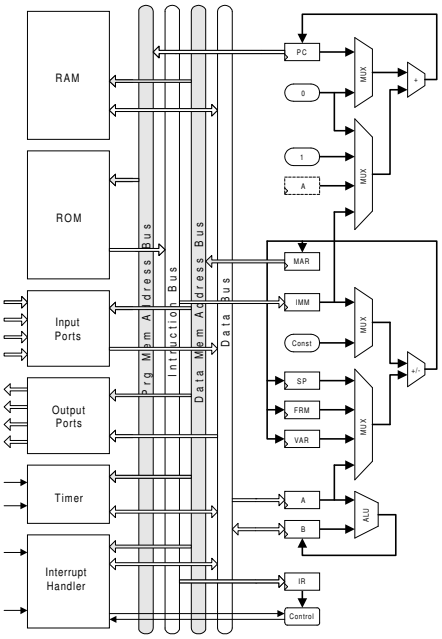

Figure 1. The FemtoJava Micro Architecture

In order to implement a scheduler, hardware support is needed. The original version of the FemtoJava microcontroller does not have instructions dedicated to

process scheduling and context switching. New extended instructions have been created for this, adding to the set of extended bytecodes to those already existing in the architecture.

The first extended instruction created was the INIT_CTX. This new instruction has the purpose to store the stack pointer value of the processes in a private memory position. These values allow the scheduler to restore the correct information from the registers, allowing the process to be executed correctly.

The second extended instruction, called INIT_STK, is the responsible instruction for the creation of a stack for each new process in the specified memory positions.

The third instruction was called REST_CTX. This instruction has the purpose to restore the process stack pointer from the position where it was stored.

The fourth extended bytecode, called SAVE_CTX, saves the current process stack pointer before being selected and placed in the queue.

Finally, the fifth instruction, called SCHED_THR, is responsible for the scheduling policy.

Table 1 presents the new developed instructions and the μinstructions number for each one of them.

Table 1. New Instruction Set

Instruction	Bytecode	μInstruction number
INIT_CTX	F4	7
INIT_STK	F5	7
REST_CTX	F6	10
SAVE_CTX	F7	6
SCHED_THR	F8	11

The set of these five new instructions allowed the scheduler development, making possible the context storage and the allocation of the processor for the competitive processes.

4. EVALUATION

To evaluate and validate the new developed instructions, a Round-Robin scheduler was implemented. Two different sort algorithms were used as application to be processed concurrently on the FemtoJava microcontroller. The Bubble Sort and Select Sort algorithms have been used to sort distinct vectors of ten elements. The software Altera Max+Plus II v10.1 and the CACO-PS simulator tool [6] was used to evaluation. The CACO-PS is a code-compiled simulator, based on clock cycle execution that calculates the consumed power in each architectural component (registers, multiplexors, and others). In accordance with the switch activity of these components, the tool informs the dynamic power consumption in switched gates (SGs). Table 2 presents the results of the scheduler impact, using the new instructions, on the embedded system.

Table 2. Scheduler Impact on the Embedded System

	Values
Needed cycles number	128 cycles
Power consumption	719.723 SGs
ROM overhead	96 bytes
RAM overhead	3 bytes
FPGA overhead	106 logic cells

Table 3 presents the overhead of execution time, number of cycles and power consumption for the Round-Robin scheduler.

Table 3. Scheduler Overhead

Algorithm	Execution total time	Executed total cycles	SGs
Bubble + Select	2.243ms	12.081	64.937.719
Bubble + Select + Scheduler	2.743ms	16.625	89.807.282

5. CONCLUSION

A instruction set was extended to support context switching was implemented on a java microcontroller and their costs were discussed. As expected, the Round-Robin scheduler introduces a smaller overhead on the embedded system. Future works will analyze the impact of different scheduling policies and will be used to create an automatic tool to synthesize embedded schedulers according to particular system requirements.

6. ACKNOWLEDGEMENTS

The authors would like to thank the support provided by CNPq, FAPERGS and PROPESQ-UFRGS.

7. REFERENCES

[1] Ortiz, S. Jr. Embedded OSs Gain the Inside Track. IEEE Computer, vol. 34, n. 11, pag. 14-16. 2001.

[2] Schlett, M. Trends in Embedded-Microprocessor Design, IEEE Computer, vol. 31, n. 8, pag. 44–49. 1998.

[3] Silverschatz, A.; Galvin, P.; Gagne, G. Applied Operating System Concepts. First Edition. Wiley. 2000.

[4] Kreuzinger, J.; Et al. Real-time event-handling and scheduling on a multithreaded Java microcontroller, Microprocessors and Microsystems, vol. 27, pag. 19-31. 2003.

[5] Ito, S. Et al. System Design Based on Single Language and Single-Chip Java ASIP Microcontroller", Design Automation and Test in Europe, pag. 703-707, Paris, France. IEEE Computer Society Press. 2002.

[6] Beck, A. C. Fo. CACO-PS: A General Purpose Cycle-accurate Compiled-code Power Simulator. 15th Symposium on Integrated Circuits and System Design. 2003.