The ARPA Project - Creating an Open-Source Real-Time System-on-Chip

Arnaldo S. R. Oliveira, Valery A. Sklyarov, António B. Ferrari

Abstract – **This paper describes the ARPA project. The aim of this project is to develop an open-source System-on-Chip model for real-time applications. The main component of the SoC is a MIPS-based RISC processor. It is implemented using a pipelined Simultaneous Multithreading (SMT) structure, which allows exploring the Instruction and Task Level Parallelism, decrease the context switching time and avoid speculative execution. Another fundamental component of the SoC is the Operating System Coprocessor, which implements in hardware some of the operating system functions, such as task scheduling, switching, synchronization, communication and timing. The proposed approach allows building high performance and time predictable processors optimized for embedded real-time systems that consume less energy than currently available superscalar processors.**

Resumo – **Este artigo descreve o projecto ARPA. O objectivo deste projecto é a concepção de um modelo aberto de um sistema integrado para aplicações de tempo real. O componente principal do sistema é um processador RISC baseado na arquitectura MIPS e implementado usando uma estrutura pipelined com suporte para multitarefa simultânea. Esta implementação permite combinar a exploração do paralelismo entre instruções de uma e de várias tarefas, diminuir o tempo de comutação de tarefas e evitar a utilização de técnicas complexas de execução especulativa. Outro componente fundamental do sistema é o coprocessador de sistema operativo, que implementa em hardware algumas das funções de sistema, tais como temporização, escalonamento, comutação, sincronização e comunicação entre tarefas. A abordagem proposta permite construir processadores de elevado desempenho, previsíveis e optimizados para sistemas de tempo real e que consomem menos energia que os processadores superescalares actuais.**

Keywords – **Simultaneous multithreading, Time-predictable processors, Operating system coprocessors.**

Palavras chave – **Multitarefa simultânea, Processadores temporalmente previsíveis, Coprocessadores do sistema operativo.**

I. INTRODUCTION

The number of transistors on a single chip has increased considerably over the last decades. The recent advances in integrated circuit (IC) technology has allowed the construction of complex Systems-on-Chip (SoC) and the manufacturing of large field programmable logic devices, such as FPGAs. FPGAs with the equivalent logic capacity of millions of gates are now available at reasonable price and are considered a strong alternative to custom ICs (ASICs), mainly due to its reconfigurability and low NRE costs. Assuming the continuous evolution of the IC technology, the next generation of processors will reach, by the year 2010, the 10 billions of transistors on a single chip. The mission of the engineers and computer architects is to find the best way to use efficiently this huge number of transistors, which obviously depends on the application domain.

The advances in computer architecture during the last decade led to the construction of very fast and extremely complex superscalar processors, employing advanced techniques to improve performance, such as superpipelining, branch prediction/speculation, out-of-order/predicated execution and sophisticated memory hierarchies. However, these techniques are also responsible for much of the power consumption and for the non deterministic performance of nowadays computers. Although, they were successfully applied to improve the performance of desktop computers, they are not appropriate for embedded real-time systems because predictable performance and in some cases low power consumption are considered important properties for this class of systems. In this paper we suggest an approach to explore the large integration capacity currently available, to build optimized, predictable and energy efficient SoCs for embedded real-time systems.

II. MOTIVATION

Embedded hard real-time systems typically consist of a set of concurrent tasks executing periodically or aperiodically with a well defined deadline. The execution is usually done on a single processor, which is able to execute only one task at a time. Therefore, a Real-Time executive or Operating System (OS/RTOS) is normally used to manage the concurrent execution of the tasks, performing the following roles:

- Schedule task execution accordingly to the predefined parameters and the established scheduling policy;

- Switch, i.e. preempt and dispatch, tasks accordingly to the computed schedule;
- Synchronize task execution and provide the adequate primitives to ensure a correct sequential behavior;
- Provide efficient mechanisms for inter-task communication.

In systems with high temporal resolution and/or a large number of (computationally intensive) tasks with tight parameters (small periods and deadlines) a high performance processor is required and a large number of context switches between tasks and the OS are performed during runtime. Context switches take time and spend energy without doing any useful work. A possible approach to improve the performance of such systems would be the utilization of an optimized processor with the following capabilities:

- Simple pipelined execution of instructions from each task, avoiding the complex techniques used currently in superscalar processors to minimize pipeline stalls;
- Simultaneous execution of several tasks eventually time sharing the same functional units. i.e. the instruction fetch stage is able to feed the succeeding pipeline stages with instructions from multiples tasks;
- Efficient hardware implementation of the OS functions using an Operating System Coprocessor (OSC);
- Reduced overhead associated with context switches.

Such a processor would reduce the number of context switches, explore both types of parallelism (Instruction Level Parallelism - ILP and Task Level Parallelism - TLP) and improve the energy efficiency without require considerably hardware complexity. The OSC based implementation of OS activities such as, task scheduling, switching, communication and synchronization also improves the system performance because they are performed in parallel with the user tasks, reducing processor load and the number of context switches. A close interaction between the OSC and the CPU core would also improve the utilization of CPU resources and increase the ability to introduce optimizations (e.g. at the context switching level).

III. OBJECTIVES

The main objective of the ARPA project is to create, implement and test a synthesizable model of a SoC optimized for hard real-time systems, as well as develop or adapt the respective compilation tools. This ambitious goal can be divided into the following parts:

- Create a processor architecture optimized for hard real-time systems. It must provide high performance and execute the tasks accordingly to the timing constraints specified during system design. The architectural optimizations must not be responsible for a considerable increase in power consumption. The use of complex performance improvement techniques which consume considerable area and power and introduces non-determinism must be avoided. To save project time, this work could be based on an existing architecture with stable development tools widely available.
- Develop an optimized pipelined implementation of the architecture employing simple techniques to improve performance and avoid pipeline stalls caused by control and data hazards. The implementation must explore different types of parallelism (ILP and TLP) and exhibit low overhead associated with context switches.
- Using an hardware description language (e.g. VHDL) and/or a high-level language (e.g. Handel-C) elaborate a behavioral model of the processor, which must be used directly for hardware synthesis. The model must be portable, i.e. independent of the target technology.

IV. RELATED WORK

Intel Corporation launched in 2001 the Hyper-Threading (HT) technology [1]. A HT-enabled processor is seen by the OS as a single physical processor acting as multiple logical processors. HT improves the performance of multitasking applications executing in a processor by dynamically multiplexing instructions from multiple tasks into the same functional units. Current Pentium 4 HT processors can execute two tasks simultaneously by time sharing the same hardware resources. The Sweden company RealFast launched in 2003 the "UltraFast Micro Kernel" real-time kernel/executive, previously named "Sierra 16" [2]. It implements in hardware some of the operating systems functions, conventionally performed in software. Task scheduling is fully implemented in hardware and the interrupt service routines are scheduled as ordinary tasks accordingly to a predefined priority. The CPU has a single interrupt request line driven by the executive and activated whenever a task switching has to be performed. The executive also provides timing primitives and synchronization mechanisms. Data exchange between the CPU and the executive is performed through the system bus, acting the coprocessor as an ordinary peripheral. In [3] it is also presented a configurable scheduler for real-time systems. The principle of operation and the interface are similar to the "UltraFast Micro Kernel".

The ARPA project is on the same research line of the referred work, but follows a different approach:

- Integrating closely the CPU and the OSC, allowing a more efficient use of the functional units and an optimized context switching procedure.
- Providing the means to dynamically select the scheduling policy and the static choice, i.e. at compile time, of memory sizes, number of execution contexts, task synchronization and communications mechanisms.

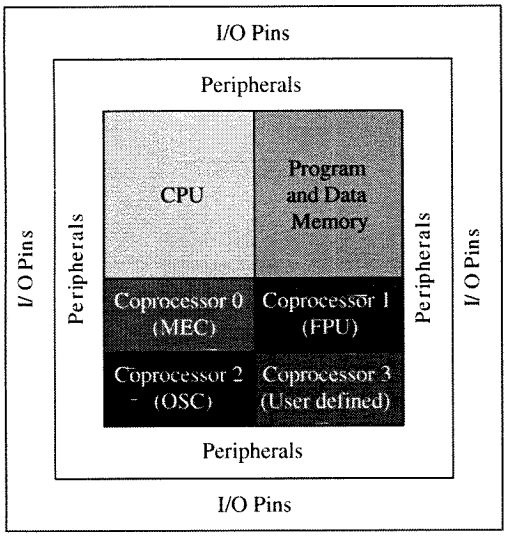

Figure 1 - Target structure for the ARPA SoC.

Comparing the ARPA project with the HT technology and the "UltraFast Micro Kernel", our approach has the advantage of simultaneously implement the OS in hardware and provide a closer interaction between the CPU and the OSC. This is possible because we are developing open models of both components.

V. ARCHITECTURE

The target structure for the ARPA SoC is depicted in fig. 1. Its main components are the following:

- The Central Processing Unit (CPU);
- Up to 4 coprocessors for exception handling, memory management, OS implementation, floating point calculations and user-defined operations;
- Program and data memory;
- Peripherals and I/O pins.

ARPA is based on the MIPS32 architecture because of its simplicity, ease of implementation and availability of stable compilation tools. In fig. 2 it is depicted the CPU pipeline. The main stages are named on the top of the figure. Depending on the implementation technology and required performance/area tradeoff, each of these stages can be split into sub-stages to bound the critical path. To support Simultaneous Multithreading (SMT), N execution contexts are provided, each supporting the execution of an individual task. Each context has its own Program Counter and bank of General Purposes Registers. The remaining pipeline components are shared by all contexts.

ARPA is an Harvard Architecture, i.e. contains separate memories for program and data. This memory organization eliminates structural hazards during memory accesses. The required duplication of the bus signals is not an issue because the CPU and memory are both integrated into the SoC. The need to predefine the memory sizes assigned to code and data is not a problem in embedded systems, because they are normally fixed at design time.

A distinctive feature of the ARPA project is the hardware implementation of the RTOS. The basic OS functions are implemented in hardware and performed in parallel with the user tasks. This approach has several advantages, such as better performance, improved predictability and requires less memory for the RTOS code and data. The main OSC blocks are shown on the bottom side of fig. 2. The *Task Table* is used to store the task parameters (entry point, state, timing information, stack size, etc.). All units have access to the *Task Table*. The *Context Switching Unit* has privileged access to the CPU pipeline. The interface and data exchange between the CPU and the OSC are based on the predefined MIPS coprocessor interface and instruction set.

VI. IMPLEMENTATION

The ARPA architecture can be implemented in different ways depending on the required performance and features. To obtain a reasonable performance/complexity compromise, the first implementation is based on a single issue, 5 stage multithreaded pipeline. To simplify and save hardware resources, all traditional superscalar techniques, such as branch prediction, and speculative/out-of-order execution were avoided. The adoption of a SMT implementation minimizes the negative impact of the proposed simplifications in the performance. Pipelined SMT allows the exploration both ILP and TLP. The ARPA SoC is being prototyped on a Trenz Electronic TE-XC2Se development system [4], which contains a XC2S300E Xilinx FPGA. The CPU core is being modeled at RTL level using VHDL because the operations performed are relatively simple and VHDL provides good control over hardware synthesis. For CPU synthesis the Xilinx XST engine is used. On the other hand, the OSC is being modeled in Handel-C because the operations and algorithms are more complex and Handel-C provides the adequate support for describing them. For OSC synthesis the Celoxica's DK2 suite is being employed. The netlists produced by both synthesis tools are integrated and used within Xilinx ISE tools for implementation on the target device.

VII. APPLICATION DESIGN FLOW

The design flow for FPGA based applications using the ARPA architecture is depicted in fig. 3. It consists of two sub-flows: the hardware design flow and the software design flow. The hardware sub-flow allows obtaining, with the aid of the synthesis and FPGA vendor implementation tools, the hardware configuration file from the behavioral SoC description. The output of the software sub-flow is an executable file generated by compilation of the software source code and used to program the system memory. If the FPGA internal memory is large enough to store the application code and data, the executable file can be merged with the hardware configuration file into a single file used to program the FPGA at system startup. The executable file

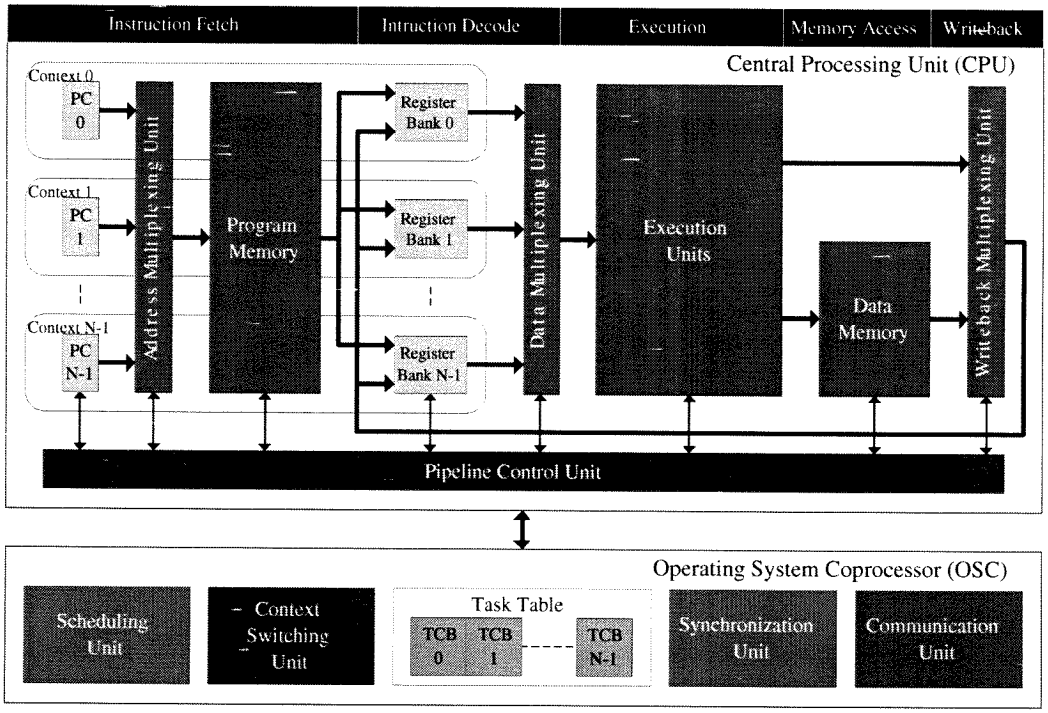

Figure 2 - ARPA CPU and OSC internal structure.

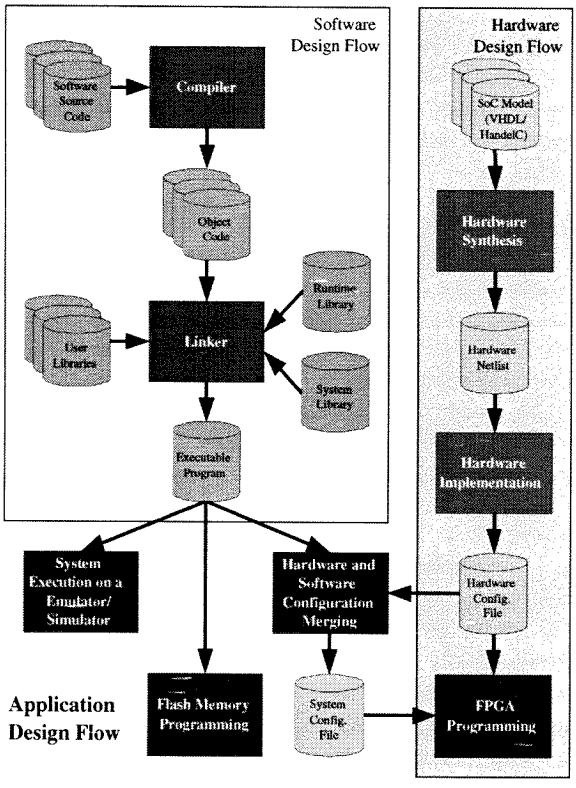

Figure 3 - Application design flow assuming an FPGA implementation.

resulting from software compilation can also be used for simulation purposes within an emulator.

VIII. CONCLUSIONS

The ARPA architecture provides a convenient platform for implementing hard real-time systems. It provides hardware support for traditional OS functions and allows exploiting both ILP and TLP. The less aggressive exploration of ILP is compensated by the TLP exploration and the hardware implementation of OS functions which allows improving performance without the complexity and power consumption of current superscalar processors. The web page for the ARPA project is at the following address: http://www.ieeta.pt/~ arnaldo/projects/ARPA/.

REFERENCES

[1] D. Marr D. Koufaty, "Hyperthreading technology in the netburst microarchitecture", *IEEE Micro*, pp. 56–65, March-April 2003.

[2] RealFast, http://www.realfast.se, *UltraFast Micro Kernel*, 2003.

[3] V. Mooney III P. Kuacharoen, M. Shalan, "A configurable hardware scheduler for real-time systems", in *Engineering of Reconfigurable Systems and Algorithms*, Las Vegas - USA, 2003, pp. 96–101.

[4] Trenz Electronic, http://www.trenz-electronic.de, *Spartan-IIE Development Platform Overview*, 2004.