Real-Time Evaluation of nMPRA CPU Architecture Based on Multithreaded Execution

Ionel Zagan*
Faculty of Electrical Engineering and Computer Science, Stefan cel Mare University of Suceava, Suceava, Romania.

* Corresponding author. Email: zagan@eed.usv.ro
Manuscript submitted October 4, 2015; accepted December 11, 2015.
doi: 10.17706/ijcee.2015.7.6.424-429

Abstract: This paper conducts a thorough study of the schedulability and predictability questions for a custom designed CPU architecture, named Multi Pipeline Register Architecture (nMPRA). The nMPRA CPU implementation uses replication and remapping techniques for the program counter, general purpose registers and pipeline registers, providing predictability and hardware-based isolation for hard real-time threads. We describe the real-time scheduling tests on nMPRA processor architecture, including also a fine-grained multithreading configuration. The present paper highlights several solutions to improve the performance of CPU architectures and to overcome the overhead inconveniences of the Operating Systems.

Key words: Fine-grained multithreading, hardware scheduler, pipeline, hard real-time, predictable.

1. Introduction

Nowadays, some of the available CPU implementations are not feasible to be used in real-time systems (RTS) with hard real-time requirements. Such systems are really critical in terms of hard real time execution, spatial and temporal isolation. In this context, even in distributed systems, timing predictability of tasks represent defining characteristics.

It is very important for a system dedicated to the control of industrial processes to be associated to a mathematical model describing as realistic as possible the practical application. A particular attention is given to the method of calculation of the processor scheduler limit with minimum temporal jitter. In order to arrive within the deadline back to the process by means of execution components, in a typical control system that interacts with the controlled process, the results produced by the tasks of the system are based on data taken from the sensors [1].

The jitter is a RTS-specific measure that may cause a deviation in the periodicity of the component tasks, threatening the stability and determinism of the application. In order to achieve this, the field-programmable gate array (FPGA) devices, as presented by Shahbazi *et al*. in [2], available with a high capacity in logic gates, can function as a hardware support for embedded real-time operating systems.

While multi-core processors outperform single-core ones, a real worst-case execution time (WCET) analysis is more difficult to obtain. The reason is that the computed execution times differ significantly from the measured WCET estimation, due to the dynamic interaction of effects between different tasks executed in multi-core processors [3]. However, in the domain of safety-related real-time embedded systems, the multi-core architectures strive to outperform single-core architectures [4], this is why the research on these

systems is increasing.

As a result, it is crucial that the most widely used system in the aeronautical and automotive field provides timing predictability behavior, with spatial and temporal isolation for hard real-time tasks. In order to improve the predictability of such systems, and to avoid the excessive use of resources, synchronization and communication mechanisms are needed to ensure a tight margin of the WCET for different tasks executed on single-core processors.

The purpose of this paper is to continue and extends the nMPRA project presented by Gaitan *et al*. in [5]. We present the experimental tests executed on a five stage pipeline architecture named nMPRA-MT (Multi Pipeline Register Architecture-Fine-grained Multithreading) [6].

At the hardware level, nMPRA-MT guarantees interrupt response with minimum temporal jitter and a minimum delay. Furthermore, the proposed CPU architecture provides predictability and hardware-based isolation for hard real-time threads. We evaluate experimentally design tradeoffs that arise when seeking to isolate tasks of different criticalities and to maintain overheads commensurate with a standard RTOS.

This paper is structured as follows: after a brief introduction located in Section 1, an overview of the nMPRA and nMPRA-MT architectures is presented in Section 2. The experimental results achieved during the tests are presented in Section 3. Subsequently, Section 4 gives an overview of the related work, while Section 5 concludes the paper.

2. Overview of the nMPRA and nMPRA-MT Architecture

2.1. nMPRA Architecture

The nMPRA architecture was designed and accomplished in VHDL language. It uses replication and remapping techniques for the program counter, general purpose registers and pipeline registers, providing predictability and hardware-based isolation for hard real-time threads.

In this pipeline nMPRA can run up to five instructions, so that the execution rate would finally reach one instruction per clock cycle.

Fig. 1 presents the nMPRA architecture involving a five-stage pipeline designed to improve computational performance and also to maintain the advantages of the pipeline.

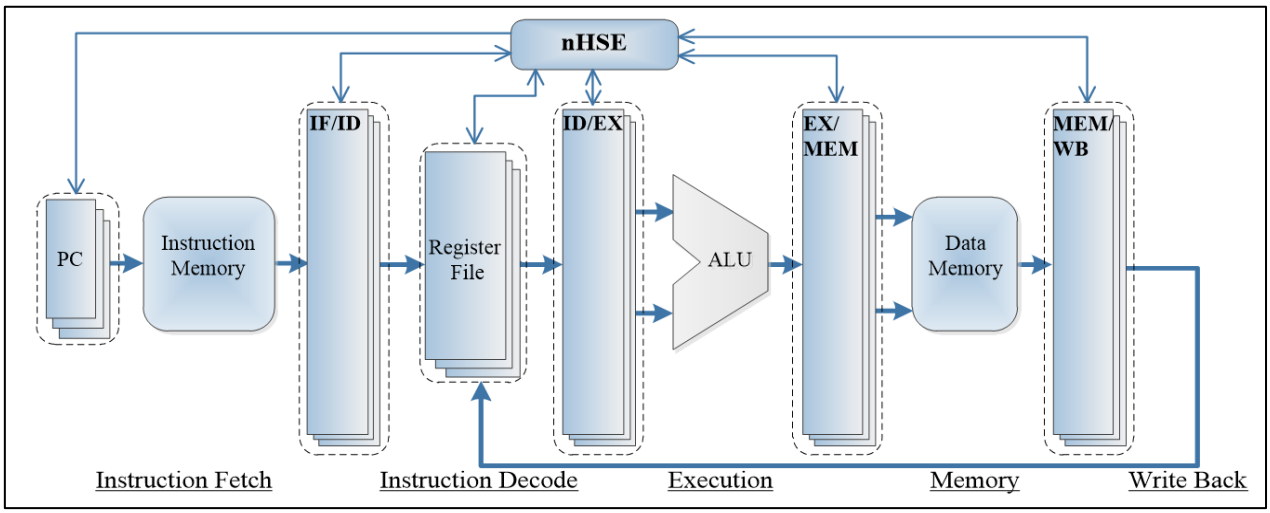

Fig. 1. Original nMPRA architecture and nHSE (hardware scheduler engine).

2.2. nMPRA-MT Architecture

The nMPRA-MT architecture was implemented in Verilog (abstract language with a syntax similar to that

of C language). The project is a continuation of the nMPRA architecture and nHSE hardware scheduler. By including in this new architecture the fundamental idea of the PRET and FlexPRET concepts [7], [8], we also make full use of its temporal repeatable behavior. For this purpose, high-priority tasks are classified and transformed into hard threads-HT and low-priority tasks into soft threads-ST.

The goal of this project was to provide predictability of hard real-time tasks, based on a fast switching operation between a blocked task and another one, scheduled by nHSE.

All the resources of the processor pipeline are shared by every thread, except for the HT0. This is the only one active after power-on or reset, having access to the nHSE configuration registers. Because a periodic and precise computation for each thread state is required, the nHSE is developed to work on the falling edge of the clock, and therefore, it is always in the active state. In order to accomplish a fast thread context switch, each HT or ST thread has a PC, a distinct bank in the register file and separate pipeline registers.

Because nMPRA-MT architecture uses resource remapping techniques for the pipeline registers and CPU working registers, nHSE interleaves different threads in the pipeline assembly line without losing clock cycles due to contexts switching operation.

In order to preserve the spatial isolation between threads, the nMPRA-MT architecture allows HTs to read anywhere in the memory, but to write only in certain areas. Moreover, the areas are specified by the nHSE that can only be set by the high priority thread. In terms of timing predictability, nMPRA-MT uses a separate memory for HT threads and a common memory space for ST threads. There is an active research concerning the memory controller, in order to obtain an efficient management and to reduce the WCET.

2.3. Pipeline and Thread Management

In order to provide both temporal and hardware-based isolation, the highest criticality tasks are assigned to HTs. nMPRA-MT architecture is equipped with a preemptive scheduler implemented in hardware that is part of the CPU itself.

In order to reduce pipeline costs, the five-stage data path was modified. As we can see in Fig. 2, the multiplexor which selects the data to be written in the register file through the MEM/WB pipeline registers is placed after the data memory. With this new solution, the MEM/WB pipeline registers have reduced dimensions and memorize only one value to be written in the next stage.

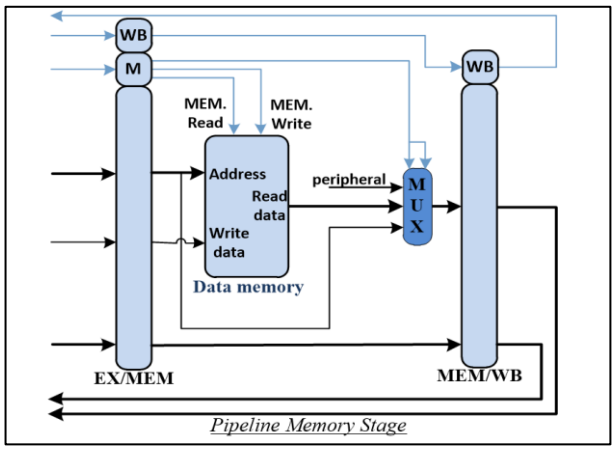

Fig. 2. Data paths improvements that reduce pipeline costs.

The dynamic nHSE is based on the remapping algorithm proposed by the nMPRA project in [5], being capable to perform fast task switching operations and to guarantee high performance in the execution of the pipeline assembly line.

3. Experimental Results

In this section, we focus on testing the presented concept, by introducing the experimental results of the implemented project as FPGA prototype. The project has been implemented on a Virtex-6 FPGA ML605 Evaluation Kit from Xilinx and the code of the processor has been developed in standard Verilog. The nMPRA-MT processor was implemented for a working frequency of 10MHz, 50MHz and 75MHz.

In Fig. 3, channel 1 from the image captured from the oscilloscope represents the clock (nMPRA_clock) used for pipeline registers and data memories. On the 2nd channel of the oscilloscope, we can see the clock waveform used for task switching in the nHSE block (nHSE_clock). This clock signal is out-of-phase with 240 degrees as compared to the nMPRA_clock signal, both signals having 33% filling factor. An asynchronous external event is shown on the 3rd channel of the oscilloscope. In this case, the event is obtained by pushing a button on the ML605 board; channel 4 marks the answer of the software application which sets a pin.

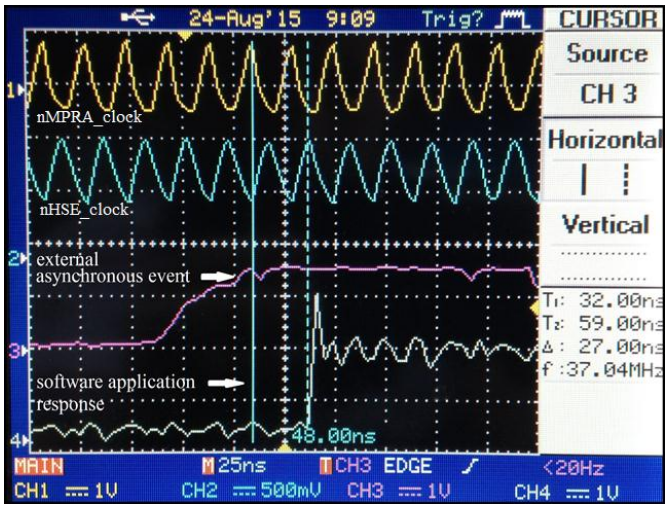

Fig. 3. Jitter of the highest priority thread HT0 in relation with an external event.

At a 50MHz frequency without synchronization and communication mechanisms, when the HT0 thread doesn't executes *sw* instructions, the answer of the scheduler to an asynchronous external event may be around 27ns, depending on the occurrence time of the event.

The processor was tested for three possible implementations of 4, 8 and 16 tasks where the maximum nesting depth of functions is on 6 levels. The approximated implementation percentage costs, relative to the FPGA device used for validation (Xilinx Virtex-6 xc6vlx240t-1ff1156), may increase with a large number of threads and with a more complex functionality of the nHSE, Hazard Detection Unit, Forward Unit and synchronization and communication mechanisms. The implementation of this architecture for a large number of tasks would call for the synthesis of a logic in which the propagation time would be groundlessly high and thus the working frequency would significantly drop.

In the pipeline processors implemented in hardware, all the logic modules operate concurrently. This means that our CPU clock must control the fair CPU structures at the required time. In order to provide the predictability of each HT execution thread, the constant scheduling frequency is required.

Table 1 presents the energy consumption for nMPRA-MT implementations with 16 tasks and a hardware support for nHSE without synchronization and communication mechanisms.

Table 1. Summary of the Power Consumption for the Proposed nMPRA-MT Implementation

Nr. of tasks:	Clocks (mW)	Logic (mW)	Signals (mW)	IOs (mW)	MMCMs (mW)	DSPs (mW)	Static power (mW)	Total mW
16	58.75	28.57	39.57	21.28	77.30	0.75	3428.6	3654.83

At a first glance, it can be said that replication of resources and remapping techniques used by nMPRA processors may represent an inefficient use of resources. However, it is important to notice that validation of nMPRA-MT architecture could provide higher throughput per area and power savings, compared to other single or multi-core systems for certain task sets.

However, for a mixed-criticality task set where predictability and hardware-based isolation guarantees for HTs are the main constraints, the nMPRA-MT processor presents a better functionality than many other traditional processors.

Due to the multiplication of resources and executing one instruction from the same thread every two clock cycles, the pipeline will never be stalled and the scheduling procedures will never have to save the context of the current task.

4. Related Work

The Merasa project [9] was developed to obtain a processor architecture which can be successfully used in hard real-time embedded systems. The priorities for execution threads are fixed and the scheduling policy chosen is round robin. Each core is made up of two scratchpad memories; one of the memories is used for data and the other for instructions (D-ISP and DSP); the data integrity is ensured by individual allocation of a subnet of banks cache for each task. Concerning the architecture, each core can have only one HRT execution thread and an arbitrary number of NHRT execution threads. Taking into consideration that the embedded systems have limited resources available, present-day architectures must offer an optimal cost for the implementation of an average number of HRT and NHRT execution threads, including their synchronization and communication mechanisms. If the HRT thread is suspended, pending an external time event or sharing a resource with another HRT or NHRT task, its dedicated assembly line will remain unused and it will negatively influence the performance of the entire system.

Therefore, although space isolation of HRT treads is guaranteed, the predictability of Merasa architecture depends on hazard situations occurring in the classical assembly lines and on the synchronization and communication mechanisms between the HRT and NHRT threads from the entire multi-core system.

In [10], Andalam proposes the ARPRET architecture, obtaining predictability by projecting a particular soft-core coupled with a hardware accelerator, called the Predictable Functional Unit. Thus, time behavior for models and programs becomes most important because, in order to guarantee that a hard real time system behaves according to the model, their characteristics must be preserved during compilation.

The aim of Komodo's project, as presented by Kreuzinger *et al*. in [11], is to use the Komodo Java-based multithreading microcontroller for handling multiple real-time threads. Thus, the architecture uses multiple stack register sets, program counters and instruction windows, and a signal unit to manage a set of threads triggered by interrupts. In order to ensure real-time support and manage multiple threads of different priorities with the proposed four-stage pipeline architecture, the picoJava instruction set is enhanced. Because the Komodo Priority manager supports fast context-switching, another thread can use the unallocated cycles in case a branch or memory access causes pipeline stagnation.

Although other threads can be executed to increase throughput, this paper does not offer a description of any synchronization or communication mechanism.

5. Conclusion and Future Work

The current paper extends the basic project presented by Gaitan *et al*. in [5], proposing original new improvements for the nMPRA and nHSE. The aim of this project was to obtain a predictable architecture dedicated to small real-time applications.

The processor architecture is subject to new improvements such as:

- Compute the WCET using complete benchmarks.
- Implement nHSE as a coprocessor.
- Increase the parallelization of the instructions execution, while maintaining the timing predictability for HT threads and efficiency of synchronization and communication mechanisms.

Acknowledgment

This paper has been prepared with the financial support of the project "Quality European Doctorate — EURODOC", Contract no. POSDRU/187/1.5/S/155450, project co-financed by the European Social Fund through the Sectoral Operational Programme "Human Resources Development" 2007-2013.

References

[1] Buttazzo, G. C. (2011). *Hard Real-Time Computing Systems — Predictable Scheduling Algorithms and Applications* (3rd ed.).

[2] Shahbazi, M., Poure, P., Saadate, S., & Zolghadri, M. R. (Aug. 2013). FPGA-based reconfigurable control for fault-tolerant back-to-back converter without redundancy. *IEEE Transactions on Industrial Electronics*, *60(8)*, 3360-3371.

[3] Ferdinand, *et al*. (Oct. 2001) Reliable and precise WCET determination for a real-life processor. *Proceedings of Conf. Embedded Software* (pp. 469–485). North Lake Tahoe, California.

[4] Agron, J., & Andrews, D. (2009). Hardware microkernels for heterogeneous many core systems. *Proceedings of International Conference on Parallel Processing Workshops* (pp. 19-26).

[5] Gaitan, V. G., Gaitan, N. C., & Ungurean, I. (2014). CPU architecture based on a hardware scheduler and independent pipeline registers. *IEEE Transactions on Very Large Scale Integration Systems*.

[6] Gaitan, N. C., Zagan, I., & Gaitan, V. G. (2015). Predictable CPU architecture designed for small real-time application — concept and theory of operation. *International Journal of Advanced Computer Science and Applications*, *6(4)*, 47-52.

[7] Edwards, S. A., & Lee, E. A. (June 2007). The case for the precision timed (PRET) machine. *Proceedings of Design Automation Conference* (pp. 264-265).

[8] Zimmer, M., Broman, D., Shaver, C., & Lee, E. A. (April 15-17, 2014). FlexPRET: A processor platform for mixed-criticality systems. *Proceedings of the 20th IEEE Real-Time and Embedded Technology and Application Symposium*. Berlin, Germany.

[9] Ungerer, T., *et al*. (2010). Merasa: Multicore execution of hard real-time applications supporting analyzability. *IEEE Micro*, *30(5)*, 66–75.

[10] Andalam, S. (2013). *Predictable Platforms for Safety-Critical Embedded Systems*. Thesis, University of Auckland.

[11] Kreuzinger, J., Marston, R., Ungerer, T., Brinkschulte, U., & Krakowski, C. (1999). The komodo project: Thread-based event handling supported by a multithreaded Java microcontroller. *Proceedings of EUROMICRO Conference* (pp. 122-128).

Ionel Zagan received the Eng. degree in computer science from the Stefan cel Mare University of Suceava, Suceava, Romania, in 2005. He is currently a Ph.D. student with the Department of Computers, Stefan cel Mare University of Suceava Romania. His current research interests include real-time systems, microcontrollers and pipeline processors with parallel execution of tasks. Mr. Zagan is a member of the IEEE Computer Society.