Real-Time Embedded Systems Co-design

Rafael L. Cancian[1] **and Marcelo R. Stemmer**[1]**, Antônio. A Fröhlich**[2]

[1]Federal University of Santa Catarina
Department of Automation and System Engineering (DAS)
Laboratory for Control and Microinformatics (LCMI)
P.O. Box 476 – 88040-900 – Florianólis - SC - Brasil
{cancian,marcelo}@das.ufsc.br

[2]Federal University of Santa Catarina
Computer Science Department (INE)
Laboratory for Software and Hardware Integration (LISHA)
guto@lisha.ufsc.br

Abstract. *This paper summarizes some techniques used in a project that goals to analyze Real-Time Systems features in a way that allows to model and to implement software and hardware components that will compose a framework repository to the automatic generation of Systems-on-a-Chip. Tools associated to that framework will allow co-design and software/hardware partitioning of such real-time features. The contribution of such project is to produce an adaptable multiplatform computational system that fulfills the real-time requirements of a particular application.*

1. Introduction

Programmable Logic Devices (PLD) have been used even often as hardware platforms for Embedded Systems (ES) and the potentiality of using configurable hardware has been recently explored to provide some kind of hardware support for Real-Time Systems (RTS). The task scheduler is considered the heart of an RTS and most of hardware support developed threads only about scheduling, although several other special features are necessary to compose a RTS. Moreover they use to be a basic support, normally restricted to queues and time management in hardware. This support may be significant for the simpler algorithms, just like Rate Monotonic (RM) and Earliest Deadline First (EDF) but is certainly insufficient for more complex algorithms and for other real-time features.

This project goals to decompose and analyze the Real-Time domain (not system) using state-of-art software engineering techniques and then to incorporate these features in a framework that allows automatic generation of application-oriented software (run-time support) and hardware (when the platform is configurable). These real-time features must be implemented in a way that allows a partitioning tool to automatically map them into software or hardware components. Knowing that the differences over hardware platforms for embedded systems are really considerable, it's necessary that such framework to provide architectural transparency and to generate systems in position to run over fixed hardware platforms (microcontrollers from 8 to 64 bits and beyond) and over configurable hardware platform, without having to rewrite application or operating system code. By doing it, this project may represent significant improvement to modeling, software engineering and co-design for real-time systems, specially for those applied to embedded systems.

This paper is organized as follow: section 2 mention some related work; in section 3 we briefly describe the actual and future development of this project and, in section 4, some considerations are given.

2. Related Work

Several hardware/software co-design or simple hardware support for RTS have been recently proposed. Hardware support for task scheduling was proposed, among others, by [Mooney and Micheli 2000], who implemented a cyclic scheduler, and by [P. Kuacharoen and Mooney 2003], who implemented priority, RM and EDF schedulers. Beyond the scheduling support [Kohout and Jacob 2003] developed hardware support for time and event management, despite supporting only fixed priority schedulers. Some other kind of support were also proposed and implemented, such as memory management [Shalan and Mooney 2000] and resources access protocols [Akgul 2003].

From the software point of view, several Real-Time Operating Systems (RTOS) for Embedded Systems (ES) are available. However, just a small part of them uses co-design. Stankovic explored in his well-known Spring system how OS components can be migred into hardware, and [Razali Jidin 2004] developed support for a multi-thread programming model that provides a transparent interface to the CPU and FPGA based components threads. The use of a unique language for hardware and software specification is interesting, but this approach has some actual limitations: in some cases the hardware algorithm is very different from the software algorithm for the same function [Grattan et al. 2002], the compilers support for just a subset of SystemC and the generated code is inefficient [Cote and Zilic 2002]. In other hand, the use of several languages in the design of embedded systems is very convenient for application development and optimization but it can become an obstacle on the way to higher design productivity, and some solutions and trends were explored by [Ernst and Jerraya 2000]

In this context, EPOS (Embedded and Parallel Operating System) is a viable alternative to became a multiplatform Real-Time Operating System (RTOS) for embedded systems. The EPOS system was born as a project to experiment with the concepts and mechanisms of application-oriented system design [AOSD] [Fröhlich 2001]. EPOS was first developed from PURE [F. Schön and Spinczyk 1998] framework of components and, through AOSD, included improvements as scenario adapters and hardware mediators that provide grate efficiency in the automatic generation of application-oriented operating systems. Later EPOS was expanded to automatically generate not only run-time software support (operating system) but also hardware support (IPs – Intellectual Properties) that fulfills the application requirements, i.e., automatic generation of Systems-on-a-Chip (SoC). This improvement is based on the concerns of abstractions and hardware mediators (from AOSD) and IPs, assigning an IP for each mediator [Polpeta and Fröhlich 2005].

3. Development

In this project, we utilize the AOSD methodology to perform domain analysis and decomposition on Real-Time systems. This enables the modification of EPOS' abstractions and hardware mediators and the development of new IPs to comply with AOSD methodology. The analysis began with the identification of related concepts and functional and non-funcional characteristics of Real-Time Systems. Based on that analysis, we are defining

families and members and their inflated interfaces [Fröhlich 2001]. These interfaces form an abstract hardware/software interface under which all hardware support for real time can be implemented, independent of the execution scenario.

Also based on that analysis, RTS domain characteristics are being included primarily in software components to make EPOS a RTOS. Because of the way EPOS was designed these real-time features can be selected/activated (or not) by an automatic configuration tool (already implemented) or by the programmer himself. Based on the inflated interface specified by the application (e.g. System calls used) and on composition rules and dependencies, the configuration tool is able to assign to each interface a member that fulfills this interface and to activate scenario aspects and configurable features.

For example, the presence of a system call for thread creation with a parameter informing its deadline (inflated interface) allows the configuration tool to infer a specific component (family member) to be selected – the member 'real-time thread'. This member may require the selection of other components, such as a member of the real-time scheduler family, a member of the resources access protocol family and the exclusion of members 'paged' or 'segmented' from the address space family. Some scenario aspects can then be applied to the selected member, such as periodicity or hardness. Although some members and scenario adapters can be automatically selected, it's impossible to infer, for example, which scheduler and task model has to be used. Therefore, some real-time members that will compose the system have to be manually selected by the programmer using a specific tool (currently implemented).

The real-time features that can be mapped into hardware or software could be described in a single language, such as SystemC. However, this approach has limitations mentioned before, and we have chosen to implement software and hardware versions in C++ and VHDL respectively. This approach shows some obvious disadvantages when compared to a single language, but it allows co-design without the actual limitations. Moreover, both versions are included into the same file facilitating comparison and maintenance. GCC preprocessor and compiler directives are used to select either one of them and to customize VHDL code, applying scenario aspects to it. After this, the VHDL code is used as input to the Xilinx tools to generate the NetList, BitStreams and finally, the ACE file, which contains both hardware and software to configure the FPGA. With this process we actually generate synthesized SoCs, and not only simulated ones.

4. Conclusions

Currently several forms of hardware support for RTS have been proposed and implemented. However, most of them are really simple when compared to the complexity of real RTS and/or are isolated proofs-of-concept in the sense that they are just fixed implementations and not the result of a RTS design methodology with co-design and partitioning. We have shown that such an approach will seldom create reusable components for different execution scenarios and then propose and develop reusable software and hardware components for a system that already allows automatic generation of multiplatform non-real-time SoCs.

We have briefly shown in this paper some strategies that are currently being used to determine and to implement real-time hardware and software reusable components for a highly adaptable new RTOS, which differs from traditional approach. Our contribution

is the use of AOSD and its techniques, as Aspect Oriented Programming, to develop and integrate hardware components (focused on real-time support) and also significant improvements to EPOS project, including real-time support, co-design and partitioning. A prototype of a generic, scenario independent hardware/software task scheduler component is included in the EPOS repository and we successfully generate functional SoCs. Finally, the conclusion of this project may represent an improvement to software engineering and co-design for real-time systems. This work was partially supported by FINEP (Financiadora de Estudos e Projetos) grant no. 01.04.0903.00.

References

Akgul, B. (2003). Hardware support for priority inheritance. In *24th IEEE International Real-Time Systems Symposium - RTSS'03*.

Cote, C. and Zilic, Z. (2002). Automated systemc to vhdl translation in hardware/software codesign. In *Proceeding of 9th International Conference on Electronics, Circuits and Systems*.

Ernst, R. and Jerraya, A. A. (2000). Embedded system design with multiple languages. In *Proceedings of the 2000 conference on Asia South Pacific design automation*.

F. Schön, W. Schröder-Preikschat, O. S. and Spinczyk, U. (1998). Design rationale of the pure object-oriented embedded operating system. In *Proceedings of International IFIP WG 10.3/WG 10.5 Workshop on Distributed and Parallel Embedded Systems*.

Fröhlich, A. A. (2001). *Application-Oriented Operating Systems*. PhD thesis, Sankt Augustin: GMD - Forschungszentrum Informationstechnik.

Grattan, B., Stitt, G., and Vahid, F. (2002). Codesign extended applications. In *Proceeding of International Workshop on Hardware/Software Codesign*.

Kohout, P. and Jacob, B. (2003). Hardware support for real-time operating systems. In *Proceedings of CODES - ISSS'03*.

Mooney, V. and Micheli, G. D. (2000). Hardware/software codesign of run-time schedulers for real-time systems. In *Proceedings of Design Automation of Embedded Systems*, pages 89–144.

P. Kuacharoen, M. S. and Mooney, V. (2003). A configurable hardware scheduler for real-time systems. In *Proceedings of International Conference on Engineering of Reconfigurable Systems and Algorithms - ERSA'03*.

Polpeta, F. V. and Fröhlich, A. A. (2005). On the automatic generation of soc-based embedded systems. In *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation*.

Razali Jidin, David Andrews, D. N. (2004). Implementing multi threaded system support for hybrid fpga/cpu computational components. In *Proceedings of International Conference on Engineering of Reconfigurable System*.

Shalan, M. and Mooney, V. (2000). A dynamic memory management unit for embedded real-time system-on-a-chip. In *Proceedings of International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 180–186.