FPGAs and Soft-Core Processors:
Understanding Computer Architectures and Processing Principles

RALF JOOST, RALF SALOMON, MATTHIAS SCHNEIDER
College of Computer Science and Electrical Engineering,
Institute of Applied Microelectronics and Computer Enegineering
Univeristiy of Rostock, 18051 Rostock, Germany

ABSTRACT
Lectures on computer architectures held at the University of Rostock indicate that a growing number of students is lacking a firm understanding of the fundamentals in the design, configuration, and programming of micro-processors. Nowadays, advanced programming techniques and design principles as well as sophisticated tools and compilers hide those basic elements. Furthermore, the faculty cannot afford to extent the classes due to the generally increasing amount of teaching content. In order to relieve these problems, this paper describes a set of practical exercises that are based on a combination of simple field-programmable gate arrays (FPGAs) and state-of-the-art soft-core processors. A noteworthy advantage of the chosen platform is that it allows for going along with the lectures by providing both a simple to use education kit for the basic parts and a sophisticated system for the cutting edge topics, such as multiprocessor systems or hardware/software co-design.

1. INTRODUCTION
The last decades have been observing a rapid development in the domain of microprocessor architectures. At the same time, remarkable changes in the field of designing-and-programming microprocessors took place. This trend significantly impacts microelectronic lectures, such as computer architectures, embedded systems, or systems design. Since the complexity of the knowledge is increasing, more and more content has to be conveyed in during the *same* lecture. Many would resort to even more sophisticated tools, which would however shift the programming and design tasks to an even higher level of abstraction. For example, hardware engineers make use of advanced design tools that allow for designing and implementing various systems by just clicking a mouse. Those tools are so sophisticated that even unskilled persons can generate advanced microprocessor system within a couple of days. This is reminiscent to the trends in the areas of programming microprocessors: The assembler language, long time state-of-the-art when programming hardware-related programs for embedded systems, has been replaced by high-level languages, such as C, C++, and Java. Although these languages provide many abstract concepts, their programs are all translated into the *old-fashioned* assembler language as an intermediate step before running on a machine of any type. Normally, most software engineers today do not care much about the target architecture, most likely because advanced compilers adapt their object code to it. This trend leads to an interesting but also problematic situation: university students who are able to design or maintain advanced systems by using various tools are on the other hand not able to

fully understand the results of their work. The consequences are dramatic: engineers are becoming more and more dependent on the tools they use, and chances for finding optimizations or improvements are low. The situation is like: “If the tool can't do it, I can't do it either”. And this is unacceptable, especially in domains that require creative minds to generate creative solutions. It is thus necessary for the students to understand all the various aspects of the processor architectures and pay attention to them. As a possible relief, this paper describes a set of practical exercises that aim at enforcing the students' self-education efforts. These exercises use the NIOSII soft-core processor [1] running on an ALTERA UP3 education kit [2] for the presentation of basic knowledge in a simplified and complete form. Further examples and explanations support the students during their work.

2. THE NIOSII FPGA PLATFORM

Teaching microelectronics and computer architectures are often limited to theoretical content, since the architectures of most processors are static by their very nature. By way contrast, the user can freely reconfigure the NIOSII soft-core processor on his/her own disposal. The NIOSII processor itself is a 32 bit RISC processor, which is designed to fit and run on a wide range of FPGAs. One of the main advantages of the NIOSII processor is that it can be configured entirely by using a graphical user interface (GUI). The soft-core approach in combination with a powerful GUI also allows for changing the processor’s internal structure and the attached peripherals as well as realizing application-specific optimizations. This possibility facilitates practical lessons of various sorts. For instance, the NIOSII processor can be used to measure how the integration of a cache memory increases the processor's computational power.

The authors have chosen the ALTERA's FPGA platform for the following five reasons: (1) it is a part of ALTERA's University Program [3] and consists of a complete tool suite as well as a low-cost FPGA education kit [2] containing the EP1C12 FPGA. (2) The tool suite ensures an easy entry in the world of microprocessor design and programming. (3) It provides specialized tools for all of the different design stages. (4) The GUI offers several tutorials that present guided tours to first skills. (5) The very same platform can be used for all the different exercises further described in Section 4.

3. LEARNING GOALS AND EDUCATIONAL CONTENT

The general idea is to provide both practical material and exercises, especially for those parts of the educational content that do not belong to the cutting edge technology. The main goals are: (1) understanding of the microprocessor's basic components, (2) understanding the principles of interplay of those components, and (3) transfer of the learning results to new and high level topics. Furthermore, the exercises aim at a wide acceptance among the students. That is, the students may do the exercises because they want to, not because they have to. In order to meet the students’ desires, the authors have asked 25 students of the computer architecture class to complete a questionnaire. The questionnaire had seven multiple choice and four open questions. According to the students' answers, the authors generated a set of

modular exercises for which Section 4 presents a detailed description.
The exercises are set up in a modular form with each one taking about an hour for its completion. Although they are organized along a central theme, parts can be skipped or rearranged as to meet the teacher's and/or students' desires. Furthermore, two or more exercises may be combined to a full-day module.
It has been mentioned above that the exercises aim at a wide acceptance. In other words, the authors want the students to realize that further studies by their own, i.e., without any external force, are of their own benefits. Thus, the exercises might be done in a self-defined order at a self-defined time. Furthermore, a unique platform is an essential prerequisite, since otherwise the students have to get acquainted to a new one for every new task. And for this very purpose, the ALTERA development tool suite is an ideal solution.

4. COURSE CONTENT

The students' answers are demanding further treatment in at least the following four relevant topics, which are part of the class and thus also relevant for the final exam:

1. Programming in an assembler language and execution principles, such as addressing modes, multiplexing, conditional branches.
2. Configuring a complete interrupt system including interrupt service routines (ISRs), the interrupt vector table, and interrupt handlers.
3. Performance improvements by special cache memories.
4. Hardware/software co-design in practice.

All exercises follow the same structure. First of all, the students have to recapitulate the class content. One of the intents is that even a plain repetition might have a positive effect. Then, they are requested to answer various questions, which provide a means for self-assessment. In so doing, the students are getting sensible for the last part, the fulfilment of practical tasks. Some selected exercises are (for a complete set, see [4]):

1. The very first exercise includes an introduction to the tools and their usage. The students are then requested to write some small assembler routines, and to compare them to compiler generated assembler code of C routines. For the purpose of a deeper understanding, the students are asked to execute their programs line by line. For this learning goal, the processor’s debugging functionality is an excellent tool, because it has read access to the processor’s internal registers as well as its memory (RAM).
2. The second exercise focuses on the interrupt system. With the help of the debugger, the students analyse the interrupt system's components, such as the interrupt vector table, the interrupt service routines, and the interrupt registers for both pending interrupts and interrupt status. After the successful completion, the students are asked to write their own interrupt service routines for given hardware elements.
3. This exercise analyses how the integration of a cache memory increases the processor's performance. This part is highly supported by the NIOS processor's dynamic configuration capabilities.
4. The last and most advanced exercise is on hardware/software co-design,

and requires basic knowledge of a hardware description language. The students have to build their own hardware components, e.g., a sophisticated logical combination, and integrate it into the NIOS processor. Finally, the students are asked to compare the performance of their hardware-augmented systems with pure software solutions.

6. EVALUATION

By the time of writing (will be extended in the final version), the authors have gone through all exercises together with 5 students (out of 25 subjects). All of them have given a very positive feedback by stating that the exercises provide a very good supplement to the regular class. They also highly appreciated that they could do the exercises at their own pace. In turn they encouraged the authors to extend the practical part of the class.

7. CONCLUSION AND FUTURE WORK

This paper has presented an approach to enforce self education of the students of the computer architecture class. This approach consists of a variety of practical exercises in combination with an FPGA education kit. The main benefit is a reusable platform that ensures both a handy interface for learning basic topics and a sophisticated toolbox for advanced topics. As a side effect, this platform approach helps reduce the costs for teaching material. Since the FPGA implements a so-called soft-core processor that can be defined and modified within a graphical environment, the students get in touch with a real processor rather than simulators or fixed hardware. In the first run, the students were very positive on this form of extra teaching. They have shown an improved understanding of the class material, and did not complain about the extra time; in their view, the benefits in terms of their grades and understandings outweigh the extra efforts. In addition, these benefits came at almost no additional cost for the teaching staff.

The authors plan to intensify the cooperation with other universities in order to create a pool of exercises accessible for all students of the participating institutes. Furthermore, integrating a web-based interface to perform the programming and maintenance of the development board shall be a concern of future work. In so doing, students can perform the practical tasks of the exercises at any place and download their results to the development board directly without the need to appear in the computer pool. The ongoing development will be directly adapted to the feedback given by the students, treating the university more in the way as a service provider for *learning* rather than a type of old-fashioned *teaching* school.

8. LITERATURE

[1] NIOSII Processor Reference Handbook, ALTERA Corp., 2005

[2] UP3-1C12 Education Kit – Reference Manual, Cyclone Edition, System Level Solutions Inc. (USA), 2005

[3] ALTERA's University Program, 2005, http://www.altera.com/education/univ/program/unv-overview.html

[4] http://www-md.e-technik.uni-rostock.de/ma/sm53/pra/main.html