Diverse Hardware Platforms in Embedded Systems Lab Courses: A Way to Teach the Differences

Falk Salewski, Dirk Wilking, Stefan Kowalewski

***Abstract*— Traditional methods for teaching the design of embedded systems usually deal with either a hardware or a software view of the system. In computer science it is mostly the software view. The hardware issues taught mostly deal with CPU based systems only and seldom with reconfigurable hardware. We recommend having a more general view at embedded systems in the way that it is always a programmable hardware platform (CPU based or reconfigurable hardware) which has to be programmed in a suitable programming language. In this context we offer a lab course where students should get familiar with different hardware platforms used in embedded systems. They should solve the same task both with a CPLD and a microcontroller each in order to clarify the differences between the two implementations. In this paper our experiences in this field of embedded systems education are described as well as our plans to continue.**

***Index Terms*— Computer science education, Lab course, Real-time and embedded systems**

I. Introduction

THE number of embedded systems is increasing. These days, already more than 98% of all microprocessors are found within embedded systems [3]. Such systems integrate hardware and software components and require developers with skills in both subjects. In this respect an interesting aspect is, which types of hardware are applied preferably in embedded systems. This information should form the basis for the education of hardware and software for embedded systems. Aside from many versions of CPU based systems like microprocessors, microcontrollers (MCU) and digital signal processors (DSP) also two other groups of hardware can be found. The first group consists of reconfigurable hardware as complex programmable logic devices (CPLD) and field programmable logic arrays (FPGA); application specific integrated circuits (ASIC) form the second group. In most embedded system design courses for computer science students only the design of CPU based systems is taught [1], [2]. However, reconfigurable hardware is a good option to improve many embedded applications [4], [8]; an option which students will probably never choose if they do not get in touch with it before. According to the need of education in this field [1], [2], [3] we aim to teach our students the general ideas of different hardware and their corresponding software. In the lab course presented in this paper the differences between CPU based systems and reconfigurable hardware are focused on.

All authors are with the Embedded Software Laboratory - Chair of Computer Science XI, RWTH Aachen University, 52074 Aachen, Germany `(surname)@informatik.rwth-aachen.de`

Since reconfigurable hardware (CPLD/FPGA) has blurred the gap between hardware and software programming [1] it has become harder to distinguish between these two areas. This is our reason to recommend dealing with all kinds of ”programmable hardware” like hardware whose behavior is defined by software. On the one hand, an MCU is used, which is programmed in classical software (C in this case) as a representative of CPU based hardware. On the other hand, we use a CPLD, which is programmed in a suitable hardware description language (VHDL in this case), as a representative of reconfigurable hardware.

The basic ideas of the differences between these two hardware approaches should be taught in a lecture. This basic knowledge should include the possibilities to implement sequential and parallel structures, possibilities of interfacing internal and external hardware modules, limitations according to the amount of memory and the number of logic cells etc. However, if the students lack the practical experience, most of them would not consider designing a system containing reconfigurable hardware. Thus, we recommend a lab course to become more familiar with these embedded systems. Since we focus on the differences of hardware platforms and their according software, a single task is given that has to be solved successively with the CPLD and the MCU. Thus, a suitable task had to be found which is representative for the field of embedded systems and which could be solved on the basis of both types of hardware. We will deal with this topic in the following chapter. Another important aspect regarding the lab-structuring is how much a priori knowledge can be assumed, which issues can be taught in a short introduction as well as how much the students can teach themselves by using the according material. It is also important which hardware and software is necessary during the lab and how the lab course should be structured. These points will be discussed in chapter 3. Chapter 4 then states the experiences we gained in the first run of our lab course and the most interesting results are presented in chapter 5. Then, chapter 6 states some advantages of this kind of a lab course and lab courses in general. We conclude with chapter 7 and present our plans for future work.

II. One task for different implementations

The task chosen for the lab course should allow presenting as many properties of embedded systems as possible. In our opinion the most important properties are the need to interface with the environment (real time requirements, concurrency) and with external peripherals (e.g. bus driver, memory chips, engine driver). Other important properties are

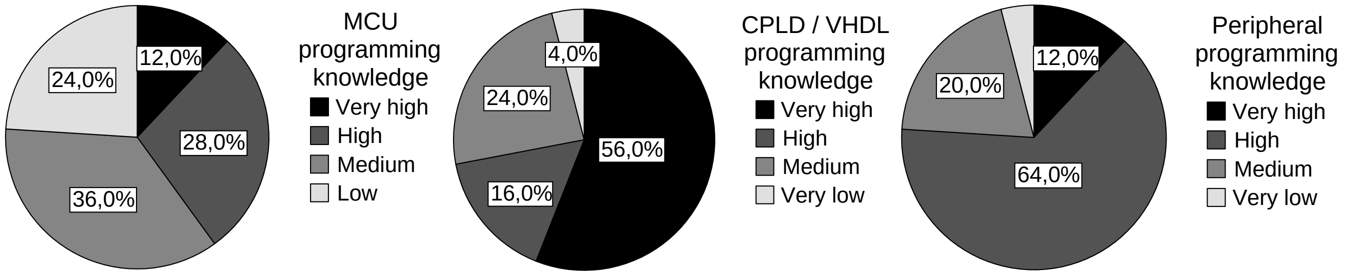

Fig. 1. Improvement of programming knowledge by hardware aspect.

the restricted resources in memory/logic cells, space, cost and power consumption. The task we chose for our lab course is the implementation of the speed measurement for an automotive prototype vehicle we are designing at our institute for research and teaching purposes [6]. The task comprises three major sub-tasks: first, the actual speed measurement, second, a measurement data processing and third, communication via CAN bus.

The sensor signal for the speed measurement is rectangularly shaped with a frequency proportional to the speed of the according wheel. Four of these signals have to be evaluated concurrently in order to gain new data for all four wheels in a short interval. Maximum values for delay and accuracy are given.

The measured values have to be processed according to a given table in order to fit into a given format of the CAN message. Furthermore an error message should be calculated.

The measured and processed values of the actual speed values have to be sent to the CAN bus as soon as new data is available in a single CAN message. For an easy access to the CAN bus an external stand alone CAN controller (SJA1000) is used which has to be initialized before.

Since this task is a combination of parallel and sequential tasks we believe that it is suited for both implementations. Details of the task can be found in [7].

III. The Lab Course

According to the fact that hands-on work significantly improves a students learning and subsequent retention of material [5], all student's should have access to suitable development boards. For a lab course consisting of 24 students twelve sets of hardware were needed (groups of two). Since all students should work on both hardware platforms they were split up in two main groups. One group started with MCUs and the other one with CPLDs. This procedure allowed working with six MCU and six CPLD boards. For the CPLD the Xilinx CoolrunnerII CPLD Starter Kit was chosen, for the MCU we used a development board based on the ATMEL ATmega16 8-bit RISC MCU. Both decisions were made according to the following reasons: low price, free development environment and good support with tutorials, news groups etc.

The development environments used are based on freeware only for the following reasons: First we would not be able to afford the high number of licenses. The second and even more important point was, that students this way could use the environment at home in order to become familiar with it. The simulator, which was available in both cases, even allows the debugging of example implementations without having the actual hardware.

For the access to the CAN bus additional boards with CAN controllers and the according bus interface were needed. We designed boards, which can be connected to the MCU (5V device) and to the CPLD (3,3V device), and provided 6 of them. We also provided 6 simple frequency synthesizers for testing the velocity measurement (simulating the sensor signals). Therefore, two groups had to share one CAN board and one frequency synthesizer for debugging.

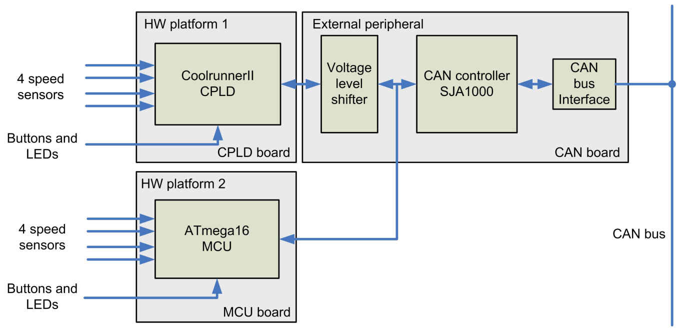

Fig. 2. Used hardware structure (use CPLD or MCU).

For the participation in this lab course we recommend visiting the lectures *Introduction to Embedded Systems* and *Embedded Software Design* offered by our institute. The first lecture deals with the basic properties of embedded systems and provides an introduction for microcontrollers and programmable logic controllers. We are planning to add basics of Programmable Logic Design in the future. The second lecture deals with development processes and methods for software for embedded systems. This includes requirements engineering (functional and non-functional requirements) and architecture design and analysis. In addition to this lectures we offered a two day introductory course in the week before the actual lab course had started. On the first day, the students were introduced to the programming of MCUs in C. On the second day, an introduction was given in the programming of CPLDs in VHDL. Since most of the students were not familiar with

VHDL, an introduction to the basics of this language was given as well. At the end of each day the same example (traffic light with user interface) had to be implemented on the actual hardware in order to give a first impression of the differences. During the introductory course we tried to clarify the important properties and techniques for MCUs (e.g. Interrupts, I/O ports, internal peripherals like timer, hardware specific C commands, debugging) and CPLDs (e.g. parallel/sequential structures, I/O ports, VHDL, debugging). Data sheets for all hardware, the introduction slides and a document with VHDL basics were handed out to each group on a CD.

The actual lab course took place on 13 appointments for 3h/week. We had to offer more appointments for some students who had not completed their task after the 13 dates yet; however some of the students also finished earlier. Since there had been no dedicated room available for the lab course, the hardware equipment and all cabling had to be set up for each day. Therefore, for installation and unistallation about 15min each were needed.

During the lab course all CAN boards were connected to a single bus, which in turn was connected to a PC based CAN bus monitor. On the monitor, the students could check if they were sending their messages correctly. The general function of the velocity measurement could be tested with the simple frequency synthesizers. For testing the accuracy of the sent values a waveform generator was available. The successful communication between the MCU/CPLD and the CAN controller could be checked by a special LED on the CAN board.

The students had to organize their work on their own and to find the necessary information in the according data sheets. However, at least one instructor was available all the time to answer upcoming questions as well as to help if necessary. Most of the answers given to one group during or after the lab were made available via Email to the other students the following day.

At the end of each implementation an acceptance test was done in order to check the basic functionality. Each group also had to hand in a documentation of their final versions (CPLD and MCU).

IV. Experience with Lab Course

The first lab course has been finished successfully with 26 students by now. Almost all of the students managed to pass the final acceptance test with both of their hardware implementations. During the introductory course most of the students learned very enthusiastically and showed strong motivation for teaching themselves the necessary additional knowledge. This knowledge included how to work with the integrated timers (MCU), the hardware specific parts of the programming language C (MCU) and the basics of the programming language VHDL (CPLD). For both implementations the access to the external CAN controller (SJA1000) and its correct use had to be understood.

According to the high number of possibilities in the field of the SJA1000 initialization many problems have occurred in this context. Wrong initialization was hard to detect since students had difficulties in separating the problems they had in the software part of the communication (wrong values in initialization, incorrect timing in write/read function, incorrect order of write/read accesses) and the hardware part of the communication (problems with bidirectional communication, incorrect jumper settings, incorrect connections in general). Thus we are planning to provide more detailed information about the use of the SJA1000 and its initialization in the next lab.

One of the major problems that occurred with the implementation on the CPLD were the restricted resources. Many students had problems fitting their design on the chip. In most of the cases this problem could be solved by adapting the calculations done in the design (reducing size of variables, simplifying calculations). However, in some cases the problem remained and thus we are planning to use bigger FPGAs (much more logic available) for the next term. We also realized that some students were not able to implement a useful combination of parallel and sequential structures successfully in VHDL. Thus, more information how to solve these problems has to be given in the introductory course next term.

Another problem concerned the CPU time needed for a compile run (synthesize, translate and fit) for the CPLDs. It turned out to be much higher than the one needed for the MCUs (minutes instead of seconds). Thus, we will try to improve the performance of our computers for the next term.

According to the longer compile runs and the less familiar software and hardware the groups which started with CPLDs needed longer to finish their first implementation. Thus, we had to provide some extra time for this implementation. This problem was solved by shifting the date of changing the groups by one week and by offering an extra date. However, we are planning to use more hardware sets in the next term to avoid the problem according to the need of changing the hardware in the middle of the course.

At the end of the course, all of the students had to fill out two questionnaires in order to evaluate the lab course. The first one is part of the university teaching evaluation and the second has been developed at our institute to improve the contents of the lab course itself. Some of the results are presented in the following chapter.

V. Results

In this chapter, the most interesting results of the evaluation will be presented. One of these results concerns the question of how much the students have improved their knowledge in the field of programming MCUs, CPLDs and interfacing external peripherals. According to the results presented in figure 1 most improvements could have been gained in CPLD and peripheral programming. However, a small number of students stated that they had learned only very little in this field. Another interesting point is the amount of external help the students needed in order to solve their task (figure 3). In the case of the CPLD significantly more external help was needed. This result corresponds well with figure 4 in which the amount of code

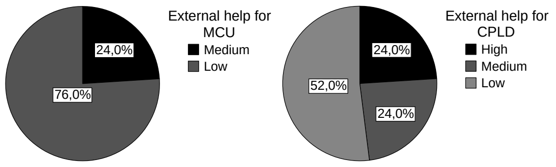

Fig. 3. External help for different platforms.

developed at home is presented. In the case of the MCU about 76% of the students never have programmed at home, in the case of the CPLD it only has been 32%. Figure 5 provides some additional information about the CPLD programming done at home with respect to the fact if the groups started with the MCU or the CPLD.

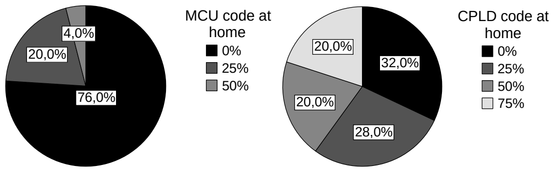

Fig. 4. Programming done at home.

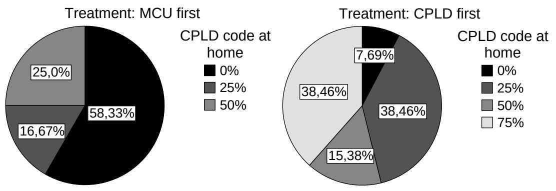

Fig. 5. CPLD programming done at home by starting group.

Those who started with the CPLD did much more CPLD programming at home than the group who started with the MCU. This could be interpreted in two different ways. First, the groups who had just started could have been more motivated and have thus worked more at home. Second, the first steps to solve the given task (understand the general ideas of CAN communication, speed measurement, etc.) were harder to realize in the case of the CPLD. In the next term an according question will be added in the questionnaire to clarify this point. Another aspect we will have to look closer at in the next term is presented in figure 6. According to this figure some students (20%) have to improve their ability to distribute the work within their group. Finally, the students were asked how useful

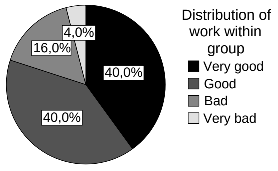

Fig. 6. Distribution of work within a group.

the knowledge gained in this lab course is to solve future tasks in this field. The result presented in figure 7 represents in our opinion a good result for this first lab course. Since many results indicated that the programming of the CPLD needed more effort we will try to reduce this difference by giving a more detailed introductory course in the next term.

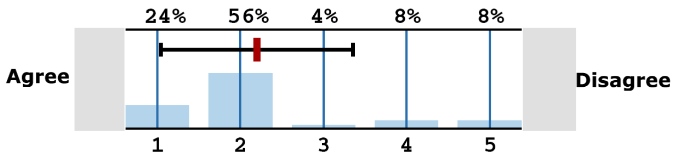

Fig. 7. Reuse of knowledge.

VI. Advantages of Diverse Platforms in Embedded Systems Lab Courses

Lab courses always involve a lot of work before and during the actual course, especially when a lot of hardware is used. However, we believe that they provide the best way to teach the practical aspects of embedded systems. Especially the use of diverse platforms turned out to be an ideal way to clarify the differences between the different platforms used in embedded systems. In our view, the results gained from this first lab course confirm this position. Additionally, a lab course could be used for collecting empirical data on many aspects of the embedded systems domain. In our case all versions of software compiled during the lab course were collected. We are planning to analyze these data in order to receive information about the differences between CPLDs and MCUs according to non-functional qualities as reliability, safety, maintainability and changeability.

VII. Conclusions and Future Works

A. Conclusions

In this paper, a way to teach embedded system design with focus on the difference of diverse hardware platforms like CPLDs and MCUs and their according languages was proposed. In this context, the same task had to be solved on a CPLD and a MCU each by every group in order to demonstrate the differences. The biggest problem that occurred during the lab was the inappropriate use of parallel and sequential structures, the limited resources in the case of the CPLD and the problems in understanding the data sheet of the SJA1000. We are sure to meet these problems in the next term by improving our introductory course and by using a larger device. We also stated that lab courses like the one presented in this paper are well suited for the collection of empirical data in the field of embedded systems.

B. Future Works

As mentioned above, the lab course for the next term will be improved by using different hardware and by extending the introductory course in the fields where the problems occurred. We are also planning to add some small additional tasks as a simple user interface. This way, we hope to improve our intention in clarifying the differences between reconfigurable hardware and CPU based systems. We also plan to implement a web page including the FAQs from the first lab course. A further lab course is planned for the summer term 2006 based on Atmels FPSLIC Chip. According to the fact that this chip includes an 8bit MCU and a 40k FPGA we hope to demonstrate not only the differences between CPLDs and MCUs but also possible approaches of HW/SW Co-Design in the way that a single task is implemented on a combination of both hardware. We plan to let the students solve a modified task on three different hardware platforms. The first one would be an MCU, the second an FPGA and the third a combination of both (FPSLIC Chip).

C. Acknowledgments

We thank XILINX (www.xilinx.com) for providing us the Spartan3 FPGA development boards we are going to use in the next term. We thank ATMEL (www.atmel.com) for providing us the FPSLIC development boards we plan to use in summer term 2006.

REFERENCES

[1] J. M. P. Cardoso. New challenges in computer science education. In *ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, 2005.

[2] N. Chang and I. Lee. Embedded system hardware design course track for cs students. In *Proceedings of the 2003 IEEE International Conference on Microelectronic Systems Education*, 2003.

[3] R. Hartenstein. The changing role of computer architecture education within cs curricula. Invited talk, Workshop on Computer Architecture Education (WCAE'04) at 31st International Symposium on Computer Architecture. http://helios.informatik.uni-kl.de/staff/hartenstein/lot/hartensteinwcae04.

[4] R. Hartenstein. The digital divide of computing. In *Proceedings of the ACM International Conference on Computing Frontiers*, pages 357–362. ACM press, 2004.

[5] A. R. Korwin and R. E. Jones. Do hands-on, technology-based activities enhance learning by reinforcing cognitive knowledge and retention? *Journal of Technology Education*, 1(2), 1990.

[6] Project:. Experimental vehicle for automotive software design. http://www-i11.informatik.rwth-aachen.de/Versuchstre+Design&bl.html.

[7] Webpage:. Lab course programming embedded hardware. http://www-i11.informatik.rwth-aachen.de/Programmierung+Eingebetteter+Hardware.html.

[8] S. Wong, S. Vassiliadis, and S. Cotofana. Embedded processors: Characteristics and trends. Technical report, Computer Engineering Laboratory, Delft, The Netherlands, 2004.