Educatng Hardware Design — From Primary School Children to Postgraduate Engineers

Oliver Knodel, Martin Zabel, Patrick Lehmann and Rainer G. Spallek
Department of Computer Science, Technische Universität Dresden, Germany
firstname.lastname@tu-dresden.de

Abstract—The future of hardware development lies in massively parallel hardware architectures as used in embedded as well as high-performance systems, for instance streaming-based, real-time and database applications. Especially field-programmable gate arrays provide a platform for the rapid development of integrated circuits and the accompanied software. For reasons of energy efficiency, it is increasingly important to tailor hardware directly to the application. As such systems are very complex, the training of engineers has to start early. Furthermore, the usual curricula in computer science and electrical engineering teach only basic skills. Our approach does not start with specialized courses in master or diploma programs, but already with the motivation of primary school children for technical disciplines. Girls and boys are addressed equally. In school, children are made familiar with programmable circuits and thus motivated to study computer science with a specialization in computer engineering. In this paper we present our lectures and practical courses for bachelor and master students. With examples from final bachelor and master projects we demonstrate the quality of our education and its integration into current research.

I. INTRODUCTION

To improve the performance of applications, multi-core and multi-threaded processors were in recent years combined with special dedicated hardware accelerators [1]. It is foreseeable that the future of hardware development lies in more and more massively parallel hardware architectures as used in embedded as well as high-performance systems. Especially field-programmable gate arrays (FPGAs) provide an energy-efficient way to achieve high performance by tailoring hardware directly to the application. The transfer of an application or an algorithm to FPGAs requires fundamental and profound understanding of reconfigurable hardware. It is necessary to translate the application to hardware level with the help of hardware description languages (HDLs). These languages have become the state-of-the-art technique in research, industry and university education.

This paper presents our concept of teaching students hardware design based on the hardware description language VHDL. It features an all-inclusive curriculum as part of the bachelor, master and diploma programs in computer science at the university. Our goal is it to prepare students for the challenges in industry or research and to motivate them to solve problems in the topic of hardware design, validation and evaluation on their own. In addition to the education of students, a concept is shown in which also primary school children and their teachers are motivated for the field of computer science. A summer school for graduate students complements the range of courses. Fig. 1 gives a summary of our education pyramid.

The following section reviews related work in the area of hardware design education. We present the children’s and teachers’ motivation in Section III. Afterwards, Section IV gives an overview of the courses for university students, followed by the additional courses in Section IV-H. In Section V we present improvements of the lectures.

II. RELATED WORK

There is only a limited number of publications on the topic of hardware design education. Particularly, there are no publications as of our knowledge, which involve children and teachers in an all-inclusive education concept. The following related work focus on graduate courses as a specialization in the curriculum.

Sagahyoon discusses a HDL-based design course and points out that HDLs are becoming more and more important in education and should be taught in undergraduate courses to serve the need of hardware design experts for industry and research [2]. He asserts that teaching VHDL is too complex and thus a simple HDL, namely AHDL, should be preferred to introduce hardware description.

Bühler et al. present a two-semester course in hardware design education in [3]. In the first semester, basic design and architecture knowledge is taught, comprising basics in VHDL, design styles and methods, evaluation and test methods, the Synopsis tool chain, and a non-pipelined DLX architecture. In the second semester, students work on a project to implement and extend a DLX processor. They also reveal the need of an experienced tutor to supervise the students.

Figure 1: The education pyramid for hardware design starts with the motivation of primary school children and extends to the research and development level.
In recent years, remote laboratories and models to access FPGAs in a server room from the workplace or even from home are introduced [4]. This concept offers the opportunity to share lab resources by time multiplexing, and to save lab equipment, space and costs in research as well as in teaching [4]. By surveying hands-on, simulated and remote laboratories Ma et al. observe that a lab with real hardware, debug tools and supervision by tutors can’t be replaced in undergraduate courses [5]. We will discuss remote labs as an opportunity for our ASE-Lab in Section V.

III. COURSES FOR SCHOOL CHILDREN AND TEACHERS

This section introduces the start of our approach at the bottom of the education pyramid (see Fig. 1). It is not our aim to instruct school children or high school students in hardware design but rather to motivate them to participate in technical subject at university. Studies show that even high school students often do not know about the work of computer scientists or they have an incorrect perception about this field of science [6]. Therefore it is important to show primary school children as soon as possible what computer science really is and to motivate them and their teachers for this field.

The children’s university gives children the opportunity to attend special lectures in the university’s lecture hall about everyday scientific topics. In this context we offer the Research Workshop Computer Science. This workshop intends to introduce children to the field of computer science and shows simple concepts in a playful way without the use of computers. For example, the children are shown the functioning of the internet by searching websites inside university buildings.

Today, teachers play a key role for the career choice of school children. School curricula typically include courses focusing on software, e.g. the use of spreadsheets and databases, or programming languages. However, to motivate for computer engineering it is necessary to provide access to digital logic. As building circuits out of standard CMOS chips is complex and costly, field-programmable logic provides an alternative. We offer an advanced training for teachers on the implementation of combinational and sequential logic in VHDL.

IV. COURSES FOR UNIVERSITY STUDENTS

This section presents the courses offered as part of the bachelor and master programs in computer science at our university. As shown in Fig. 2, students can earn bachelor’s degree after six semesters (three years) and a master’s degree in four additional semesters. Alternatively, students can enroll in a combined program to earn a diploma degree after five years. The teaching methods in bachelor and master courses include lectures, exercises and labs. Each course is completed by either a written or a verbal exam, or by a presentation and a defense. The students get a number of credit points which are valid in the European Credit Transfer System (ECTS).

Fig. 3 illustrates that our education is dominated by exercises and labs (63 % in total). Thereby, students learn to apply their theoretical knowledge to real world problems. The bachelor project and the master thesis are not assigned to a specific method because they are organized by the students themselves.

A. Computer Engineering for Bachelor Students (CEBS)

This course is the starting point of our education path and lays out the fundamentals for studies in the area of computer engineering. Students learn how digital systems work from a technology perspective, including necessary foundations of electrical engineering such as the basics of electric and magnetic fields followed by the introduction of the main electric components and the computation of electrical
networks. Because of their growing importance, electrical and optical wires are explained as well. The next part covers basics on semiconductors, OpAmps and CMOS logic. Finally, based on CMOS gates, the students learn to analyze and synthesize combinational and sequential logic. Besides lectures and exercises, selected topics are practiced in a lab (CE-Lab).

B. Computer Architecture (CA)

This course consists of two parts. The first part covers the structure of processors and how calculations are performed, whereas parallel systems and HPC architectures are discussed in the second part. Our part (1st one) starts with the several numeral systems used by computers and a revision of boolean algebra, combinational and sequential logic. Next, the execution of software is illustrated based on machine code and the von-Neumann architecture. Our part is concluded by several improvements to the bottlenecks of this architecture.

C. Programmable Logic Devices (PLD)

This course gives an overview on computer aided design of integrated circuits (ICs) to attract bachelor students to the modules of the master program. The FPGA-based design of ICs is explained employing VHDL. Its concepts and syntax are introduced by simple combinational and sequential logic and its mapping to FPGAs. In contrast to Sagahyroon [2] we consider the transition from writing imperative software to parallel hardware description the greatest challenge for the students. In the VHDL lab, students get familiar with the Altera Quartus-II toolchain and learn to implement simple decoders, arithmetic, counters and finite state machines (FSM) and connect them to larger designs. The correct functioning of the designs is checked by simulation as well as programming it on the Cyclone-III FPGA (Terasic DE0). The course concludes with an overview of current debug and trace methods for ICs.

D. Computer Engineering for Master Students (CEMS)

The CEMS lecture is the starting point for all lectures in the computer engineering path of the master program. Students are introduced to the design, modeling, programming, simulation, and implementation of technical systems from three perspectives: VLSI system design, embedded systems, and parallel processing. In our part (1st one) the students get familiar with system architectures and modeling paradigms of VLSI systems. Moreover, they are enabled to verify descriptions of hardware systems by simulation and convert these descriptions to real circuits using typical design tools. The lecture starts with an introduction to circuit design with a special focus on the interaction of FSMs. The architecture of FPGAs is explained alongside with high-level modeling approaches such as VHDL. The lecture continues with topics such as timing analysis and computational power vs. power efficiency. The knowledge is deepened by a lab. Starting from the implementation of simple decoders, counters and FSMs, the final task is to design a stop watch with split time memory. The functionality has to be verified by simulation and is tested on the DE0 board.

E. VLSI Design of Circuits and Systems (VDCS)

Building on CEMS, this course deals with problems which arise from designing complex systems and presents possible solutions. The lecture discusses problems of a broad range, which include VLSI design styles, innovative programmable SoC architectures, low-power design, redundant systems, crossing of clock domains, and hardware implementation of several arithmetic functions. Aspects from the digital as well as from the analog domain are covered. The lecture is accompanied by a lab in which students practice the implementation of their own system. Again, VHDL and a FPGA prototyping platform (Spartan-3 Starter Kit) are used to implement a scientific calculator or simple games such as snake, mancala, minesweeper, or boxworld. Their common feature is that, at first, controllers need to be designed which handle the input and output to external components. Their functionality is checked against predefined hardware test benches by simulation as well as on the FPGA board. Afterwards, the computing or gaming logic has to be designed and verified. The lab is concluded by the simulation of an analog circuit. Especially, the analysis of the transient behaviour of typical CMOS gates gives a better understanding of the maximum switching frequency of ICs.

F. Advanced System Engineering Lab (ASE-Lab)

The knowledge from previous courses is combined and deepened to build one’s own large hardware architecture which uses ALUs, interconnected FSMs, pipelining, or caching. The main challenge in this lab is the understanding and the implementation of the pipelined control and data flow. Especially, control hazards such as pipeline stalls and reverses must be treated. Also the critical path must be optimized to reach a high clock frequency. Beside verification by simulation, the communication with external components such as Gigabit Ethernet requires the use of on-chip logic analyzers.

One possible task is to build a SoC with a CPU, I/O components, and on-chip memory. All components must be interconnected by two Wishbone buses (Havard Arch.) to easily allow the addition of multiple slave devices and the reuse of existing components. The CPU consists of a five-stage pipeline including a forwarding unit [7]. Only a subset of the 32-bit MIPS instruction set must be supported, including ALU ops, branches and jumps, as well as load and stores. C-programs can be compiled with the GCC tool chain. In contrast to Bühler et al. [3], the final design can be tested on real hardware.

Another available task is the implementation of a streaming-based UDP/IP stack in hardware utilizing a data flow architecture with appropriate interfaces. The implementation includes the UDP layer with checksum calculation, the IP layer and the integration of the ARP protocol. Apart from the low-level Ethernet core, memories, FIFOs and a test bench are provided. The test bench realizes a loopback on the MAC layer so that the UDP/IP stack can be verified by simulation. Functional tests as well as the measurement of bandwidth and latency are conducted by C or Java programs sending UDP packets.

G. Final Projects – Bachelor Project, Master Thesis

A share of about 20-30% of the course participants choose the field of hardware design for their master or diploma thesis. Fig. 4 shows the number of participants in our courses and the number of final projects. The theses require a presentation on the current state of research, a motivation for the approach chosen and an evaluation of results. Compared to these, the complexity of bachelor projects is considerably smaller.
Many final projects consist of extensions or optimizations of a Java bytecode multi-core system-on-a-chip architecture [9]. In these cases the challenge lies in becoming familiar with the complex Java bytecode architecture and improving a part of it. Examples are the implementation of a scheduling and loading distribution and performance tuning by the integration of a data cache. Another kind of final projects consists of the analysis and the development of communication protocols. For example a streaming-optimized interface for a SATA controller was implemented as a final bachelor project [10].

Furthermore, there are projects focusing on solving computationally intensive tasks on FPGAs. These include mathematical problems, logic-based combinatorial games and scientific applications. The difficulty lies in understanding the algorithm, identifying potential starting points for parallelization and transferring the application to the register transfer level. Examples include a Sudoku solver [11] as well as a solver for the N-Queens puzzle [12]. A scientific application implemented on a FPGA is for instance the short-read mapping problem [13].

H. Additional Courses – Summer School

In addition to the courses presented above we offer a summer school in embedded systems for master and PhD students. It continues over two weeks and addresses primarily students from other domestic and international universities. The idea is it to give an overview of state-of-the-art embedded systems and reconfigurable hardware. In the first week the students get an overview of hardware design on the level of the CEMS lecture introduced in Section IV-D. The goal is to familiarize the students with the modeling paradigms of VLSI systems. In the second week the students learn the basic principles of hardware/software codesign and solve a task in a lab on a programmable system on chip (PSoC).

V. CONCLUSION AND OUTLOOK

This paper presents our approach of a comprehensive hardware design education. Due to the complexity of hardware system design we argue that the training of engineers has to start as early as possible in the bachelor courses rather than in the specialized courses of master programs. Based on our experiences hands-on labs with actual hardware are of utmost importance for understanding the process of hardware design. Furthermore, examples of final projects demonstrate the integration of courses and current research.

The main insight gained from the various labs is that it is important to simulate the designs as long as possible because the implementation and the debugging on real hardware are too complex. Providing test benches which simulate real hardware as correct as possible strongly support the students’ work. The provision of VHDL test benches to generate the input and check the output improves efficiency through automation and simplifies supervision. The final transfer of the design to real hardware is necessary to demonstrate the complete process of hardware development as also observed by [14].

The advanced labs like VDCS, ASE-Lab and the final projects often require neither direct access to the hardware nor the physical presence of the students. By providing a 24/7 remote lab, an easy and flexible sharing of the resources between multiple students and the remote access to FPGA resources can optimize the workflow. Similar to [4] we employ multiplexing to save lab equipment, space and costs in research. To provide remote access to the FPGAs we plan to create an infrastructure containing some of our powerful FPGAs [15].

REFERENCES