An Undergraduate Processor Design Course with Hardware-Software Collaboration

Qiang WU¹, Man-man PENG¹ and Jian-jiang Li²,*

¹College of Information Science and Technology, Hunan University, Changsha, China
²Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing, China

*Corresponding author

Keywords: Higher education, Teaching reform, Course teaching, Processor design course.

Abstract. Microprocessors are critical to the information technology. Learning processor design is essential for undergraduate students of computer science and engineering related majors. It is challenging to teach undergraduate students processor design in a limited time. In this paper, we introduce our experience of teaching the processor design with hardware-software collaboration. In addition to hardware circuits, students also learn to design the assembler, loader, and simulator to form a close-loop design flow for processor design. With this hardware-software collaborative approach, the students can complete the course tasks and achieve the course goal more efficiently.

Introduction

ACM curricula recommendations include computer architecture and organization as one of the core knowledge blocks for computer science and engineering majors [1, 2]. Universities and colleges that offer undergraduate programs on computer science and engineering provide processor design or equivalent course as the focus of the computer architecture and organization knowledge block [3, 4, 5, 6].

Since processor design involves not only hardware, but also software, some teachers enhance software participation in processor design courses [7]. Exploiting simulators to illustrate basic principles of the processor and other computer components is a common way in teaching processor design or computer architecture and organization [8, 9, 10].

While simulators are good at helping students understand the knowledge, it is a passive way for learning. So some teachers propose to enforce active hands-on practices by utilizing the reconfigurable hardware platforms [11, 12, 13, 14].

Our idea is to keep this hardware-software collaborative approach to teaching processor design, but reduce the complexity and comprehensiveness a little bit for undergraduate students. We have applied the hardware-software collaborative approach in processor design course for two years. The experience we have got is outlined as follows:

- A course plan of design tasks suitable for undergraduate students;
- A set of software tools feasible for the students to develop;
- A close-loop design flow built with the software tools.

The paper is organized with this section as an introduction. Section 2 introduces the course plan and the related processor design flow. Section 3 describes the experience of our course conduction. Section 4 concludes the paper with an outline of the future work.

Course Design

Course Tasks

The goal of the processor design course is to strengthen students understanding of computer architecture and organization, especially microprocessors, through designing a relatively simple
processor including single-cycle and pipelined versions by their own. For both versions, students are required to complete the processor circuits design, as well as the related software tools including assembler, loader and simulator.

**Course Plan**

Students of the processor design course have to finish hardware circuits design task as well as software tool tasks including assembler, loader and simulator. Hardware circuits design is the core task while software tool tasks are supporting tasks. The plan for carrying out these tasks is outlined in Figure 1.

The idea behind this plan is to form a close-loop design flow with the software supporting tools, facilitating the design and verification of the processor hardware circuits. The details of the close-loop design flow with the relations between design tasks are shown in Figure 2.

The design starts from processor specification and assembly program for testing. Assembler and loader transform the assembly program to memory initialization file in the format of Verilog memory data.

Hardware circuits design is done with Verilog. The test bench reads memory data file generated by the assembler and loader, produces outputs of the processor in terms of register and memory states. The outputs are compared with the reference outputs generated by the simulator from the assembly program. If they match, the processor design is verified and completed. If not, debug should be done for the processor design as well as the assembler, loader and simulator.
Teaching Experience

Overview
We implement the processor design course in our universities through 2016 to 2017. Students who take the course all reflect that this course helps them to turn the knowledge on the textbook to real estate in their mind.

A further discussion with the students reveals that writing the simulator and the close-loop design flow help them most in understanding the operating mechanism and hardware design conventions of the processor.

Simulator Design
Apparently, the simulator for single cycle processor is easier to design than that for the pipelined processor. In fact, we receive more questions from the students when they begin designing the simulator for single cycle processor because the students have no idea how to design a simulator at the very beginning.

Through deep investigation, we find that the primary difficulty for most students is about modeling the behavior of the processor. Then we tell the students to focus on the machine state of the processor, which means the values of 32 general purpose registers and program counter, as well as memory locations involved in the execution of the assembly program. For the pipelined processor, the internal registers for pipeline operation should also be taken into consideration. After our explanation, the students start rolling the ball of simulator design and raise less questions in later steps.

Close-loop Design Flow
In our previous teaching experience, students find the processor design verification difficult due to the lack of handy supporting tools. One of the reasons is because the processor for teaching purpose is always different from those in the market which may have sufficient commercial supports but too complicated to be used in education.

To balance the complexity and comprehensiveness, we choose to let students develop relatively simple software tools for their processor circuits design. It can be seen that a close-loop design flow is formed by the software supporting tools including assembler, loader and simulator in Figure 2.

Responses from the students show this hardware-software collaborative approach to processor design is more efficient than what the students experience in before. With the close-loop design flow, the students are able to iterate the design process time by time, improving the design towards the final acceptance. In addition, since the software tools are developed by themselves this time, the students have nearly full control of all the tools in their hands, leading to a systematic, or “bird-eye” view of the whole project, which helps “a lot” in completing the processor design task, as remarked by some of the students.

Summary
This paper introduces our plan and implementation of a processor design course for undergraduate students. Our idea is applying a hardware-software collaborative approach to the processor design course. With software tools developed by the students, including assembler, loader and simulator, a close-loop design flow is built for the hardware circuits design and verification. Responses from students show effectiveness of this hardware-software collaborative approach.

Nevertheless, this processor design course is run only for 2 years. We are making further revisions to the course plan and implementation details according to the students reflections for a better acceptance of the course in the future.
Acknowledgement

The work is supported by the Education Reform Project of the Hunan University (531111000002) and the Construction Project of the Research-oriented Teaching Demonstration Courses of the University of Science and Technology Beijing (KC2017YJX17).

References

[1] Association for Computing Machinery. (2016, December). Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering (CE2016). Retrieved August 21, 2017, from https://www.acm.org/binaries/content/assets/education/ce2016-final-report.pdf.

[2] Association for Computing Machinery. (2013, December). Curriculum Guidelines for Undergraduate Degree Programs in Computer Science (CS2013). Retrieved August 21, 2017, from http://www.acm.org/education/CS2013-final-report.pdf.

[3] Lillian (Boots) Cassel, Deepak Kumar. (2002). A state of the course report: computer organization & architecture. Proceedings of the 7th annual conference on Innovation and in computer science education (ITiCSE '02) (pp. 175-177). New York, NY: ACM.

[4] X. Liang. (2008). A survey of hands-on assignments and projects in undergraduate computer architecture courses. Advances in Computer and Information Sciences and Engineering (pp. 566-570). T. Sobh, Netherlands: Springer.

[5] E. Larraza-Mendiluze, N. Garay-Vitoria. (2015). Approaches and Tools Used to Teach the Computer Input/Output Subsystem: A Survey. IEEE Transactions on Education, 58(1), 1-6.

[6] S. Ristov, N. Ackovska, V. Kirandziska, D. Martinovikj. (2014). The significant progress of the Microprocessors and Microcontrollers course for computer science students. Proceedings of 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 818-823). Opatija, Croatia: IEEE.

[7] D. Kehagias, M. Grivas. (2009 December). Software-oriented approaches for teaching computer architecture to computer science students. Journal of Communication & Computer, 6(12), 1-9.

[8] B. Nikolic, Z. Radivojevic, J. Djordjevic, V. Milutinovic. (2009 November). A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization. IEEE Trans. Educ., 52(4), 449-458.

[9] R. Hasan, S. Mahmood. (2012). Survey and evaluation of simulators suitable for teaching for computer architecture and organization Supporting undergraduate students at Sir Syed University of Engineering & Technology. Proceedings of 2012 UKACC International Conference on Control (pp. 1043-1045). Cardiff, UK: IEEE.

[10] G. S. Wolffe, W. Yurcik, H. Osborne, M. A. Holliday. (2002 February). Teaching computer organization/architecture with limited resources using simulators. SIGCSE Bull., 34(1), 176-180.

[11] X. Wang. (2011). Multi-core system education through a hands-on project on fpgas. Frontiers in Education Conference (FIE) (pp. F2G-1-F2G-6).

[12] J. H. Lee, S. E. Lee, H.-C. Yu, T. Suh. (2012). Pipelined cpu design with fpga in teaching computer architecture. IEEE Transactions on Education, 55(3), 341-348.

[13] N. Ackovska, S. Ristov. (2013 March). Hands-on improvements for efficient teaching computer science students about hardware. IEEE Global Engineering Education Conference (EDUCON), (pp. 295-302), Berlin, Germany: IEEE.

[14] A. T. Markettos, S. W. Moore, B. D. Jones, R. Spliet, V. A. Gavrila. (2016). Conquering the complexity mountain: Full-stack computer architecture teaching with FPGAs. Proceedings of 11th European Workshop on Microelectronics Education (EWME) (pp. 1-6), Southampton, UK: IEEE.