AC 2007-1279: IMPLEMENTING A HANDS-ON UNDERGRADUATE COURSE IN SOFTWARE-HARDWARE CODESIGN

Yi Cheng, California State Polytechnic University-Pomona

Kathleen Hayden, California State Polytechnic University-Pomona

Zekeriya Aliyazicioglu, California State Polytechnic University-Pomona

Implementing a Hands-On Undergraduate Course in Software-Hardware Codesign

Yi Cheng, Kathleen Hayden, Aliyazicioglu Zekeriya
California State Polytechnic University, Pomona

1. Introduction

The Electrical and Computer Engineering (ECE) Department of California State Polytechnic University, Pomona started to offer a Bachelor of Science of Computer Engineering in fall, 2003. This Computer Engineering program offers a balanced curriculum in both software and hardware; there are seven quarter courses in digital hardware, and seven courses in software. These courses are taught in a traditional way; the interaction and trade-off between hardware and software design is hardly covered in any computer engineering courses. The faculty members have been trying for several years to integrate hardware with software courses.

The ECE faculty members have been working with the managers and engineers of the Department Industrial Advisory Council to update our curriculum. With their encouragement, we started to teach hardware-design language and digital design based on Field Programmable Gate Array (FPGA) in our Electrical Engineering and Computer Engineering programs. They also told us about their strong interests in the applications of software-hardware co-design (SHC) in embedded systems. With their assistance, we were able to set up a digital design laboratory with design software from Cadence, Synopsis and Xilinx.

There are many inexpensive FPGA boards from several vendors, which offer flexibility, performance and ease of use. Our students can design, implement and test digital circuits using a hardware description language, Verilog or VHDL. Many FPGA boards allow users to instantiate a high performance processor, such as MicroBlaze, PowerPC, and ARM, etc., and its memory and I/O devices on an FPGA chip. Therefore, we can use the board to implement a very flexible and powerful embedded system. Furthermore, these boards can also serve as platforms to explore the SHC system design and testing. We decided to change an existing junior-level course in IBM-PC Interfacing [2] to an introductory course in SHC, and offered it in fall 2006.

2. SHC Lecture and Laboratory Course

This junior-level hardware-software co-design course consists of a 4-unit lecture and a one-unit laboratory. Students attend a 65-minute lecture three times a week, and a 3-hour laboratory course. The pre-requisites of this course are the two digital logic courses, which teach basic logic analysis, design and simulation using Verilog design, two C/C++ programming courses, and one introductory embedded system course.

2.1 Lecture Course

The lecture course consists of three components: the MicroBlaze computer system architecture, embedded system software design using MicroBlaze Device Driver

Application Program Interface (API), and an introduction to the SHC. It was originally planned to devote half of the course to cover different aspects of SHC. However, only one third of the class time was actually spent there because our students needed more lectures to learn the MicroBlaze system and its device driver API.

The students studied the MicroBlaze processor architecture to get familiar with its registers, memory organization, addressing modes and instruction set. The busses, including Local Memory Bus, On-chip Peripheral Bus and Fast Simplex Link, were studied in details, because our students needed to learn to interface their hardware modules to these busses, and the bandwidths of these busses might limit the performance of a SHC system.

Teaching the embedded software using the device driver API took a lot of class time. It was not enough to teach the API functions and offer many example programs. A student needed to study the hardware block diagrams and descriptions of each I/O device before he/she could really understand how to use software to initialize and control these I/O devices. Due to its importance in real-time processing, the timer was the most studied I/O device. The timer module has two 32-bit counters, which have three modes of operations: generate, capture and pulse-width-modulation (PWM). Our students were able to implement a PWM controller, or an input capture function to measure the period or frequency of an input signal and generate real-time interrupts.

We used the book [1] to provide reading material for the SHC. We discussed the typical design flow of SHC: partitioning, scheduling and implementation. Among the three co-design classifications: SHC of embedded systems, SHC of instruction set processors, and SHC of reconfigurable systems, we studied only the SHC of the embedded systems. We used several embedded systems as case studies: a car dashboard system, a real-time spectrum analyzer using FFT calculations, MP-3 encoding and decoding, and MPEG encoding and decoding. We compared the performance a software-only implementation with that of a SHC implementation, whose hardware handled the repetitive computation-intensive tasks of calculating FFT, encoding or decoding. Even though our students had not studied the theoretical aspects of the algorithms of the MP-3 or the MPEG, they were highly interested in the different implementations of these algorithms.

2.2 Laboratory course

Students were able to implement a base embedded system with a Xilinx MicroBlaze 32-bit processor by using the Xilinx Platform Studio in the laboratory. The system consisted of a MicroBlaze processor, an instruction and data memory connected to the Local Memory Bus, and the various I/O devices that are connected to the On-chip peripheral bus. The I/O devices included two general-purpose 32-bit I/O ports, one timer, an interrupt controller, a UART, and a customized hardware module to interface with an LCD. They learned to write embedded software in C by using the MicroBlaze Device Driver API. After compiling their programs and downloading their design to their Xilinx Spartan-3 FPGA boards, they tested their systems.

They could also add interfaces to different I/O devices by adding IPs (intellectual properties) from the Xilinx library, including general-purpose I/O ports, UARTs, an interrupt controller, timers, or they could write hardware modules in Verilog or VHDL.

Students normally used Xilinx Integrated Software Environment (ISE) to synthesize their hardware modules. With a SHC system, the MicroBlaze processor and its I/O devices were converted to their Netlist for the ISE to synthesize. This is the Co-synthesis process.

They could also simulate a SHC system, or Co-simulate, by using a test fixture to provide input stimuli for both the MicroBlaze and the hardware modules. The ModelSim program by Mentor Graphics was used to generate simulation timing diagrams to validate a SHC system. The simulation outputs provide very detailed, clock-by-clock results. We also needed higher-level simulation outputs, which can be tied to the C statements and variables. However, the Xilinx and ModelSim simulation programs do not provide them.

Debugging of a complex SHC system was made easier by using Xilinx ChipScope, which was similar to a logic analyzer for hardware debugging of digital circuits. A ChipScope Core was instantiated and connected to the on-chip peripheral bus to enable it to capture any data on the addresses, data, and control information on the bus. One could set up triggering conditions based on the address, data or control.

The six laboratories were adopted from a two-day faculty workshop conducted by Xilinx. Despite the minor differences in the software version of Xilinx Platform Studio and the targeted FPGA boards, most of the students in the class were able to successfully complete the lab assignments.

Several SHC design projects were studied and analyzed in the class, but not actually implemented due to limitation of time. We designed a 32-bit microcontroller with analog interface, PWM, input capture, and parallel and serial interface. We also designed a car dashboard system using the MicroBlaze system.

We are currently looking for additional SHC tools to speed up the SHC design process. A tool that estimates the execution time of a C code fragment is very useful in the partitioning decision. A program that converts a C code into a Verilog or VHDL code can provide a quick prototyping for hardware.

3. Student Learning Outcomes Assessment

The twenty-seven students in this first SHC class were highly enthusiastic and motivate to learn and work in the laboratory. Many of them thought highly of this course to include it in their resumes.

Weekly quizzes were given every Friday to assess students' learning of the topic covered in the lecture and lab during the week. The instructor adjusted his lecture according to the quiz results. A comprehensive three-hour written examination was given at end of the

quarter. Most students were judged to have learned the course material enough to apply it to their design work in the future.

The students learned the design tools very quickly. They felt that the Xilinx Platform Studio, ISE, ChipScope and ModelSim were easy to use. Their difficulties were to understand and fix the error and warning messages generated by these programs.

The students also offered several ideas to improve this SHC course. They thought a SHC textbook would be easier for them to understand the topics better, or a more complete reference material about the MicroBalze hardware and software would make their learning easier and more efficient. They also felt that an advanced digital design course would prepare them better to design hardware modules and to partition tasks among hardware and software.

4. Conclusions and Recommendations

Our first attempt to teach SHC embedded systems in a one-quarter junior-level course was a partial success. They were very enthusiastic and motivated to learn this new technology. The students learned to implement a simple embedded system using a 32-bit MicroBlaze system on a Xilinx Spartan-3 FPGA board. Due to the limitation of class time and instructor's experience, more sophisticated SHC projects were analyzed, but not implemented. We expect that class projects will be improved in the future.

A SHC course requires a student to have sufficient experiences and expertise in both hardware and software design. Our Computer Engineering program has a required course in software engineering, which is the last in the software sequence. Maybe, an instructor can teach SHC material in this class and the students can participate in a hardware-software design team project in the laboratory.

An introductory textbook in SHC will make the students' learning much easier, especially at the undergraduate level.

5. References

[1] G. De Micheli, R. Ernst, W. Wolf, "Readings in Hardware/Software Co-Design", Morgan Kaufmann Publishers, Academic Press, 2002.

[2] M. A. Mazidi and J.G. Mazidi, "The 80x86 IBM PC and Compatible Computers", Second Edition, Prentice Hall, 1998.

[3] Xilinx, "MicroBlaze Processor Reference Guide", Embedded Development Kit EDK 8.1i, October 5, 2005.