Using FPGA-based Configurable Processors in Teaching Hardware/Software Co-design of Embedded Multiprocessor Systems

Xiaofang (Maggie) Wang
Dept. of Electrical and Computer Engineering
Villanova University
800 Lancaster Avenue
Villanova, PA 19085
Email: xwang@villanova.edu

***Abstract*—With the growing popularity of using multiprocessors in embedded systems, teaching multiprocessors in universities is becoming more and more important. FPGA-based configurable processors offer a low-cost, fast turnaround, and versatile design and implementation platform for this challenging task. This paper presents an introductory embedded systems design course that integrates configurable processors, FPGAs, and multiprocessor design and implementation. We emphasize hands-on hardware-software codesign skills by a series of carefully designed tutorials, lab assignments, and a comprehensive project on the Altera DE2 FPGA development board. Developing and teaching such a new course with no appropriate textbook available is quite challenging. The course has been popular and well received as reflected by the students evaluation and feedback.**

I. Introduction

Most embedded systems are designed around one or more processors. Roughly 98% of 32-bit processors are actually being used in embedded systems [1]. In the last few years, the power and scaling challenges in deep submicron CMOS technology have forced a major industry turn towards chip multiprocessors from single-core processors. Chip multiprocessors are becoming ubiquitous in all computing domains. Embedded systems in many performance-sensitive application domains, such as military and aerospace applications, consumer electronics, and wired and wireless communications devices, are increasingly turning to multiprocessor platforms to provide better and faster services.

On the other hand, recent technology breakthroughs of field-programmable gate arrays (FPGAs) have helped them become more popular for system implementation. The advent of soft configurable processors targeting FPGAs adds fuel to this trend. Embedded system designers often find themselves struggling to choose an optimal balance among many opposing factors. Configurable processors allow users to customize and extend the feature set of the processor to address specific design issues and trade off speed, area, and power. Compared to using both an FPGA and a separate microprocessor chip on a board, integrating soft processors with other modules in an FPGA offers a single-chip system solution, reduces the development cost, and improves the system performance in terms of both latency and power. Moreover, the software design environment provided by FPGA vendors can be used to quickly configure and implement a processor-based embedded system within a very short time.

It follows naturally that embedded systems education should address this new design platform, *i.e.*, multiprocessor on FPGAs, in classrooms. Such systems benefit most from a systems perspective that emphasizes hardware-software codesign. In this paper, I present an embedded systems design course (ECE 8448) that I designed and have been teaching at Villanova University. This is an introductory computer engineering course in our Master's program. In contrast to the simulation-based approaches for learning multiprocessor [2]–[4] where the software is the primary focus, our course employs the Nios II configurable processor and the DE2 FPGA development board from Altera to provide a hands-on platform for students to actually implement their own multiprocessor design and mapping applications on their own systems. We aim to provide a comprehensive coverage of embedded systems design, with a focus on software-hardware codesign skills under various contradicting constraints. A challenging and open-ended project integrates multiprocessor design, configurable processor trade-offs, and FPGA design together and offers a lot of opportunities for students to perform trade-offs in both hardware and software at various levels to achieve good performance.

II. Course Overview

Table I shows the topics covered in the lectures. Each topic is covered in 2.5-3 hours. In addition to the lectures, students have homework & lab assignments and a comprehensive project, which will be described in detail in Section V. Students are assumed to have learned digital/logic design, computer architecture, and C programming from their undergraduate courses. Experiences in VHDL, Verilog, FPGAs and associated CAD tools are very helpful. Lab assignments are done using students' own time. A web site [5] has been used to host the course material and maintain fast and efficient communication with students. Students can find lecture slides, homework & lab assignments, project assignment, references through this web site. Students can share questions/problems and solutions through the web-based discussion forum.

TABLE I
TOPICS OF THE EMBEDDED SYSTEMS DESIGN COURSE

Week	Topics
1	Course overview; Introduction to embedded systems
2	Embedded processor architecture
3	Project kickoff; Build & implement a Nios II system on DE2 board
4	Design programs for embedded systems
5	Program compilation and optimization techniques
6	Operating systems for embedded systems
7	Real-time scheduling policies
8	Interfacing peripherals to processors
9	Introduction to multiprocessors; Design and implement a multi-core Nios II system
10	Design programs for multiprocessors
11	Conventional buses for embedded systems
12	Networks on Chip
13	Multiprocessor debugging and optimization
14	Project presentation/demonstration/discussion
15	Exam; Project report and design files due

After extensive exploration and being disappointedly convinced that there was no single textbook that satisfies our needs, I developed my own course material, including lecture slides and notes, handouts, and tutorials by integrating materials from several books, vendor technical documentations, and my own experiences as an engineer. I want students to explore hardware architectural choices to achieve various performance goals. Most embedded systems books use ARM or other fixed logic or hard processors as examples. If I had adopted an ARM development board for the course, there would be not much space for students to play with the hardware other than programming processors and peripherals, which most often they have learned from their undergraduate microcontroller courses.

I strongly believe that hands-on lab assignments and projects are essential for effective learning of engineering subjects. In order to achieve this goal, the Altera DE2 FPGA board and Nios II configurable processor are employed in our course to enable students to explore and implement repeatedly their own hardware and software design ideas. Without this platform, it would be almost impossible for our students to modify their own multiprocessor hardware design and have it running on a real hardware board in minutes.

III. Software and Hardware Platforms

The Nios II processor is a 32-bit pipelined general-purpose RISC microprocessor with separate instruction and data-memory masters (Harvard memory architecture). It comes in three flavors: fast (Nios II/f), standard (Nios II/s), and economy (Nios II/e), with different resource requirements and for different application scenarios. External peripheral devices are connected to Nios II cores via the Avalon Switch Fabric bus, which consists of point-to-point master to slave connections and supports multi-mastering transactions.

After extensive research and experiments, I have chosen the Altera's DE2 Development and Education Board for the course because of its versatility, low cost, and plenty of support resources. This board is used throughout the course for students to work on lab assignments and the project. The DE2 board features a Cyclone II EP2C35 (672-pin package) FPGA chip and is equipped with most commonly used components in embedded systems, including 512 KB of SRAM, 8 MB of SDRAM, and 4 MB of flash memory, SD memory card slot, and a full range of I/O interfaces including USB 2.0 (type A and type B), line in/out, microphone in, PS/2 mouse or keyboard port, 10/100 Ethernet, RS232, video in, and video out. These features expose students to a wide range of topics through carefully designed lab assignments and projects. This versatile board is also used in another graduate-level VHDL/Verilog-based FPGA digital design course that I regularly teach. Many other universities have also been using it for a wide variety of courses, *e.g.*, [6], [7]. Prof. Hamblen at Georgia Tech maintains a DE2 resource web site [8] that contains many design examples for various projects.

Altera Quartus II provides a GUI application called SoPC (System on a Programmable Chip) Builder to configure and build a Nios II system containing Nios II processor cores, peripheral devices, and memory interfaces. Students can implement a Nios II system within minutes by simply selecting the desired functional units and specifying their parameters. SoPC Builder generates either Verilog or VHDL files. The files then can be used to integrate other modules, *e.g.*, IP cores purchased from third parties, to implement a complex and powerful multiprocessor system on FPGAs. The Nios II IDE is an integrated environment in which users can create, modify, build, execute, and debug Nios II programs in C/C++ and assembly languages.

IV. Tutorials and Lab Assignments

As seen from the last section, this course involves several complex development tools. A series of tutorials are designed and provided to students to help them get familiar and start using these tools.

Tut1: Build a Nios II hardware system in the SOPC Builder.
Tut2: FPGA design and implementation using the DE2 board.
Tut3: Get started with Nios II IDE.
Tut4: Build a multi-core Nios II system in the SoPC builder and implement it on the DE2 board.
Tut5: Using Nios II software development tools in the Command Shell.

The following labs, in addition to theoretical problems, are used as homework assignments to enhance what students learn from the lectures and to prepare them step-by-step for the project. These labs are done outside the class meeting time, *i.e.*, using students' own time.

Lab 1
- Install the design tools and set up their license on their own PC. Most often, students prefer to work on their own laptop until they need the DE2 board.
- Read the first three chapters of the DE2 User Manual. Install the DE2 control panel and experiment with the control of the peripherals on the DE2 board from the control panel.

Lab 2
- Configure and implement a Nios II system with a 5-bit output PIO (Parallel I/O) port and connect it to the LEDG0-4 on the board. Add another 2-bit input PIO port to the Nios II system and connect it to the SW0-1 on the board.

- Modify the "hello_world.c" to include the control of these PIOs.
- Test both hardware and program on the DE2 board.

Lab 3
- Write a C program for multiplication of two 100 $\times$ 100 integer matrices. Use the timestamp functions in the C program to count the number of clock cycles the program needs. Run the program on the Nios II system developed in Lab 2. The on-chip memory is not enough and students have to use the SDRAM as the main memory for the Nios II processor.
- Apply computer architecture knowledge to improve the Nios II configuration to reduce the number of clock cycles needed by the program.

Lab 4
- Apply the loop optimization techniques learned from the lecture to the matrix multiplication program and retime the new code on the best Nios II configuration from the last homework. Run the improved program on the DE2 board and take a snapshot showing the number of clock cycles.
- Using the direct-register access approach, light two LEDs from the program: one before the start of and one after the multiplication core. Use the green LEDs that have been added for Lab 2.

Lab 5 Students are asked to install uLinux for their Nios II system on the DE2 board and to exercise various features of the operating system. A USB drive is used to load a picture to the system and display it on a VGA monitor.

Lab 6 Exercise interrupting and direct register control of peripherals.
- Add the button KEY3 and the six 7-segment LEDs, HEX5 to HEX0, on the DE2 board as PIO (Parallel I/O) peripherals to the Nios II system.
- Write a C program to continuously show the current time (Format: HOUR:MINUTE:SECOND) on the six 7-segment LEDs. Show the hour on HEX5-4, the minute on HEX3-2, and the second on HEX1-0. Use KEY3 as the toggle button to start (or resume) or stop the real-time updating of the time.

Starting the ninth week, students focus on their project and there is no more lab or homework assignments.

V. The Project

A. Project Assignment

Many embedded systems are multiprocessors or multi-core processors involving hardware-software codesign, so is our project. The project is to design and implement a multi-core processor using at least four Nios II cores and multiply two 100 $\times$ 100 integer matrices on the quad-core multiprocessor. The final design is implemented on the Altera DE2 FPGA development board. At the end of the semester, all groups present/demonstrate their approach, results, and lessons learned, in 10-15 minutes. Specifically, there are two parts in the assignment:

1. **Hardware**: design and implement a quad-Nios multiprocessor targeting students' own parallel matrix multiplication algorithm developed for the second part of the project.

Design and implementation of multiprocessors involves complex design processes and prohibitively high design and manufacturing costs. The advent of soft core configurable processors coupled with FPGAs has provided exciting opportunities for embedded systems designers. Using the SoPC builder, students can easily modify and tune the multiprocessor and implement it again in the FPGA. During the experiments, students love the fact that they can easily explore their architectural/microarchitectural ideas, including interconnection schemes of peripherals and processor cores, on real hardware, which is impossible if a conventional development board with a processor chip is used. However, they quickly realize that developing efficient programs and managing the cores become the most difficult tasks.

2. **Software**: design and implement a parallel algorithm in C/C++ to multiply two 100 $\times$ 100 integer matrices on your quad-core processor.

The objective is to minimize the total execution time of the matrix multiplication on the quad-core processor. Good performance comes from good designs in both software and hardware, or a good balance/match of them. The parallelizing technique, *i.e.*, how to divide and coordinate the job among the cores, also has a big impact on the performance. Matrix multiplication is a fundamental kernel in many numerical algorithms. This application is a good start for students to learn how to divide a problem and design a simple parallel version of it. Given the simplicity of our hardware and lack of operating-system support for the multi-core processor, many sophisticated techniques do not bring positive returns on our system. So students focus on efficient coordination and communication of the processor cores in their multiprocessor system. I choose 100 $\times$ 100 matrices because they are larger than the capacity of the on-chip memory so students have to use on-board memory to hold their data and programs. The size of the matrices is also manageable so that students can have some intuitive insight when developing their own parallel algorithms. Students start with analyzing the sequential matrix multiplication algorithm on one processor. Then they try to extract independent steps and map them to different processor cores. They make a significant effort to balance the workload of the multiple cores to reduce the parallel execution time.

B. Suggested Steps for the Project

Students generally perform well if a challenging problem is broken down into small manageable problems. The following steps are suggested to students with an associated schedule. After the project kick-off during the third week, students start planning and working on their project in addition to their homework assignments. A brief weekly progress report is required from each group.

1) Study the sequential matrix multiplication algorithm.
2) Learn necessary information on the Nios II processor.
3) Build a single-core Nios II processor and run "hello-world.c" on the board. A tutorial on the DE2 board is provided to the students. I will help students through this step in the lab.
4) Build a quad-core Nios II processor and run the parallel version of "hello-world.c" on the board. First, I give a lecture on the introduction of multiprocessors. Then, students use a multiprocessor tutorial on the DE2 board

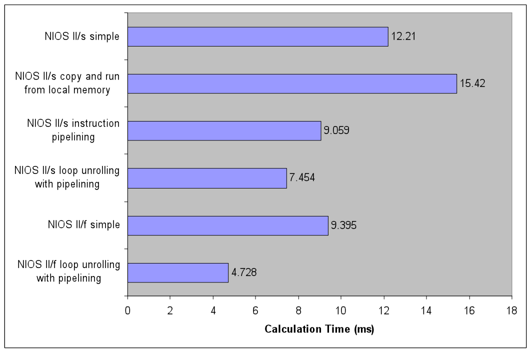

Fig. 1. Performance comparison of various hardware/software configurations of the Nios II multiprocessor system.

to get the general picture of the multiprocessor design tools. After the ninth week, students show different efforts and progress each week.

5) Propose a parallel matrix multiplication algorithm and write programs for each processor. Learn how to distribute and manage the available shared memory among the processors. Students are on their own for the parallel matrix multiplication algorithm. Generally, students have shown wonderful creativity at this part.
6) Test and tune the parallel multiplication algorithm on their own quad-core processor. Students need to demonstrate that the final result is correct and show the total number of clock cycles their algorithm takes.

Through this project, students learn how a multiprocessor works both at the hardware and software levels. They also have to learn how to manage and allocate limited resources to achieve good performance. It is manageable for an entry-level graduate student and exposes students to many aspects of software/hardware codesign of a small parallel system. Students are offered many opportunities to explore various hardware/software techniques by taking advantage of the flexibilities of configurable processors and FPGAs to optimize their design and try to achieve the best performance. The most difficult part of the process is a system workflow and coordination protocols among processors. Since this is the first time that most students are exposed to design and implementation of a multiprocessor system, most of them feel overwhelmed by the things they have to read and think through. One of the challenging problems is how to divide the on-board SDRAM among the four processors and how to manage simultaneous access to the same SDRAM chip. On-chip memory is in short supply and they have to decide how to use the limited on-chip memory to serve as a buffer/cache for the matrix data. They also have to carefully design their program in order not to exceed the capacity of their memory system. The most important advantage of the project is the opportunity to explore hardware/software codesign. As an example, Fig. 1 shows the performance results of various hardware/software configurations of the Nios II multiprocessor system designed and implemented by one students group. The best design has achieved a speedup of 3.708, which makes the system efficiency to be 92.71%.

VI. Student Feedback

Students are required to write a report about their project design, implementation, and experience. They also fill out an anonymous teaching evaluation form at my absence. The key questions and their scores are listed in Table II, which shows the course was very well received by the students. Students generally wrote that it was a comprehensive and very intensive course. The project was challenging but exciting and required a lot of team work to complete in the allocated time. During the process, they also learned a lot of lessons, such as how to divide work among team members, how multiprocessor programming for embedded systems is different from programming desktop computers, *etc*. Some students came into this course with no knowledge of FPGAs and exited it with hands-on skills to work on FPGA-based projects.

TABLE II
Fall 2009 Student Learning Evaluation of ECE 8448

Question	Score (out of 5)
Organizes and plans course effectively	4.6
Hard work is required to get a good grade	4.3
I found the course intellectually stimulating	4.3
I learned a great deal in this course.	4.3
Rate overall quality of instruction	4.4
Rate overall value of this course	4.4

VII. Conclusions

In this paper I described my experience in designing and offering an introductory course on embedded systems design. To reflect the trend of increasing popularity of multiprocessors and FPGAs in embedded systems, this course employs hands-on tutorials, labs, and a comprehensive project that involve both hardware and software design and implementation to expose students to many key aspects of embedded systems and help students develop professional skills. With limited time, students learn how to perform hardware-software codesign of an FPGA-based multiprocessor and parallelize applications to the multiprocessor. Developing such a new course with no textbook available is challenging, but it is very rewarding that the course has been very well received by the students. We hope more and more embedded systems courses will teach multiprocessor hardware and software as they enter the mainstream of embedded computing.

References

[1] W. Wolf and J. Madsen, "Embedded systems education for the future," *Proceedings of the IEEE*, vol. 88, no. 1, pp. 23–30, Jan. 200.
[2] O. Ozturk, "Multicore education through simulation," in *IEEE Int. Conf. Microelec. Syst. Educ.*, July 2009, pp. 9–11.
[3] A. Veglis, C. Barbargires, and A. Pombortsis, "Teaching performance evaluation of multiprocessor architectures with Mathcad and MathConnex," *IEEE Trans. Educ.*, vol. 45, no. 3, pp. 231–237, Aug. 2002.
[4] A. Hansson, B. Akesson, and J. van Meerbergen, "Multi-processor programming in the embedded system curriculum," *SIGBED Rev.*, vol. 6, no. 1, pp. 1–9, 2009.
[5] "Villanova ECE 8448: Embedded Systems Architecture." [Online]. Available: http://care.ece.villanova.edu/courses/ECE8448/ECE8448.htm.
[6] "Cornell ECE 5760: Advanced Microcontroller Design and SoC." [Online]. Available: http://instruct1.cit.cornell.edu/courses/ece576/.
[7] "UIUC ECE 527: SoC Design." [Online]. Available: http://courses.engr.illinois.edu/ece527/mp2.html.
[8] "Altera DE2 Board Resources for Students." [Online]. Available: http://users.ece.gatech.edu/~hamblen/DE2/.