A Project-Based Embedded Systems Design Course Using a Reconfigurable SoC Platform

Daniel Roggow, Paul Uhing, Phillip Jones, and Joseph Zambreno
Electrical and Computer Engineering
Iowa State University, Ames, IA, USA
{dlroggow, pfuhing, phjones, zambreno}@iastate.edu

***Abstract*—Embedded systems are becoming increasingly complex, as typical system components, such as sensors and other specialized processors, are blended together with more traditional microprocessors to form complex systems-on-chips (SoCs). Teaching undergraduate students to understand concepts and technologies behind embedded systems is important in order to prepare these future engineers with the skills and expertise necessary for designing such complex systems. This paper describes an undergraduate course designed to introduce students to embedded system design concepts and challenges in an engaging and effective manner. Our course uses a combination of in-depth laboratory assignments and topical lectures to provide a unique hands-on education for students. Laboratory assignments utilize Avnet's ZedBoard platform, a development board built around Xilinx's Zynq-7000 SoC, and require students to solve a variety of embedded system challenges from a range of application domains. Overall, student feedback about the course has been positive.**

I. Introduction

Embedded systems are becoming more complex as a greater number of constituent components are combined into systems-on-chips (SoCs). Systems utilizing these devices often blend traditional boundaries between hardware and software, requiring system designers and engineers to be versed in a variety of domains in order to have sufficient knowledge of concepts and principles for creating adequate solutions. To train such knowledgeable engineers, it is imperative that electrical and computer engineering degree programs offer challenging undergraduate courses to help prepare graduates for the diverse and difficult domain of embedded systems design.

This paper outlines an undergraduate course developed to address the need for a hands-on embedded systems design curriculum utilizing current technologies. Laboratory assignments are coordinated with the lectures, and focus on solving challenges associated with embedded systems design from a variety of applications, ranging from small unmanned aerial vehicle control to image processing. By using the Avnet ZedBoard development platform containing a Xilinx Zynq-7000 SoC, students are able to gain experience as the course progresses with increasingly complex configurations of peripherals, without having to learn to use additional development platforms designed for specific applications. The course is an elective for juniors, seniors, and graduate students, and is potentially a second or third course in an embedded systems progression. Although geared towards electrical and computer engineering students, enrollment is open for qualified students from other engineering majors.

II. Related Work

Field Programmable Gate Arrays (FPGAs) are an enabling technology for application-specific systems, providing a means for rapid system prototyping and evaluation, as well as algorithm acceleration. Because of their flexibility, many undergraduate programs are beginning to teach embedded system design concepts using FPGAs, and some even provide a sequence of courses to expose students to different aspects of embedded system design. For example, the Institute of Microelectronic Systems (Hannover, Germany) couples an FPGA prototyping course with an ASIC design course to provide a 2-semester introduction to embedded systems [1]. Each course focuses on building a single project by breaking it down into different hardware units. Drexel University offers a similar sequence, but uses a programmable SoC as the target hardware platform [2].

The Rochester Institute of Technology has created a more rigorous embedded systems sequence consisting of three courses designed to advance the students' proficiency in embedded systems concepts [3]. The first course covers advanced computer organization topics, the second focuses on creating custom intellectual property (IP) components and implementing custom instructions, and the last course is a capstone course. Each course utilizes FPGAs as the target platform for laboratory and project assignments. At the Georgia Institute of Technology, students in electrical and computer engineering are exposed to embedded systems programming in the first C/C++ course [4]. Other universities provide embedded systems courses that use FPGAs to emphasize particular applications, for example, embedded multiprocessor systems [5] or graphics processing units [6]. The National Chip Implementation Center (Hsinchu, Taiwan) offers three short courses on embedded systems using different hardware platforms targeted for different application areas [7]. The Rey Juan Carlos University (Madrid, Spain) provides an in-depth embedded systems course for students near the end of their undergraduate education that covers topics relating to analog electronics, digital electronics, and computer fundamentals [8].

Many of the above courses are project-based courses, as is the course described in this work. This means students complete in-depth laboratory projects with significant problem-solving components designed to expose them to real-world embedded system concepts and challenges. The course described in this paper is one of several embedded systems courses offered at Iowa State University and is a reworking of the original CprE 488 course [9, 10].

978-1-4799-9915-6/15/$31.00

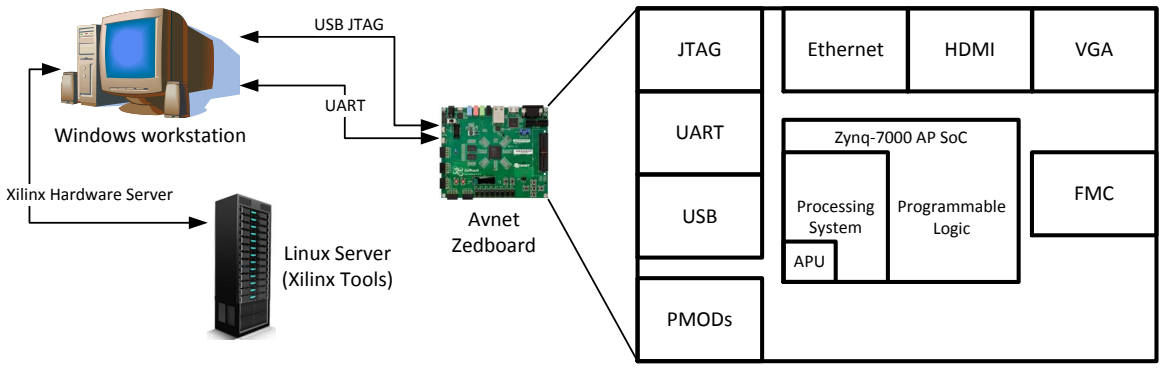


Fig. 1: The laboratory workstation configuration and ZedBoard block diagram.

III. Laboratory Resources

A. Workstation Setup

Fig. 1 shows the workstation setup for the course, and the general workflow for a machine problem (MP) is as follows: students log in to the Windows machines and start the Xilinx hardware server, which communicates with the Linux server and the ZedBoard through JTAG and UART connections. In order to use the Xilinx toolchains, students connect to departmental Linux servers using NoMachine's NX Client. Development takes place on the Linux servers through the virtual desktop provided by NX Client. After designing and writing the code to solve each MP's particular problem, students program the ZedBoard using the hardware server. Students have the opportunity to interact with a variety of peripherals using diverse communication mechanisms: JTAG, UART, USB, VGA, HDMI, and Ethernet are all built-in to the ZedBoard. Other peripherals can be attached through the ZedBoard PMOD interfaces to allow additional communication using Bluetooth or Wi-Fi. The PMODs and the FPGA Mezzanine Card (FMC) interface can also be used to connect peripherals such as oscilloscopes and digital cameras. The specific peripheral hardware utilized is described in Section IV.

B. Laboratory Hardware

The computational core of the Avnet ZedBoard (call-out of Fig. 1) consists of a Xilinx Zynq-7000 All-Programmable SoC. The three major components of the Zynq core are: the Application Processor Unit (APU), the Processing System (PS), and the Programmable Logic (PL). The APU contains a hard-IP dual ARM Cortex-A9 MPCore CPU using the ARMv7 instruction set architecture (ISA). The APU also provides support for the Thumb-2, Jazelle RCT, Jazelle DBX, VFPv3, and NEON ARM ISA extensions. The PS contains the interconnect logic to connect to the I/O peripherals, memory interfaces, and PL. The PL is where third-party or custom IP cores can be implemented as memory-mapped or streaming peripherals.

C. Laboratory Software

Students use the Xilinx Embedded Development Kit (v14.7) toolchain to develop the hardware and software portions of the lab. The Xilinx Platform Studio (XPS) is the main hardware development tool, and provides a GUI for editing Zynq configuration settings, as well as adding and configuring custom and third-party IP cores. Additionally, XPS can launch a user-specified HDL simulator for HDL files. XPS also provides the tools to synthesize and generate a bitfile for the Zynq SoC, and constructs the necessary bootstrap firmware to configure and start the SoC on power-up.

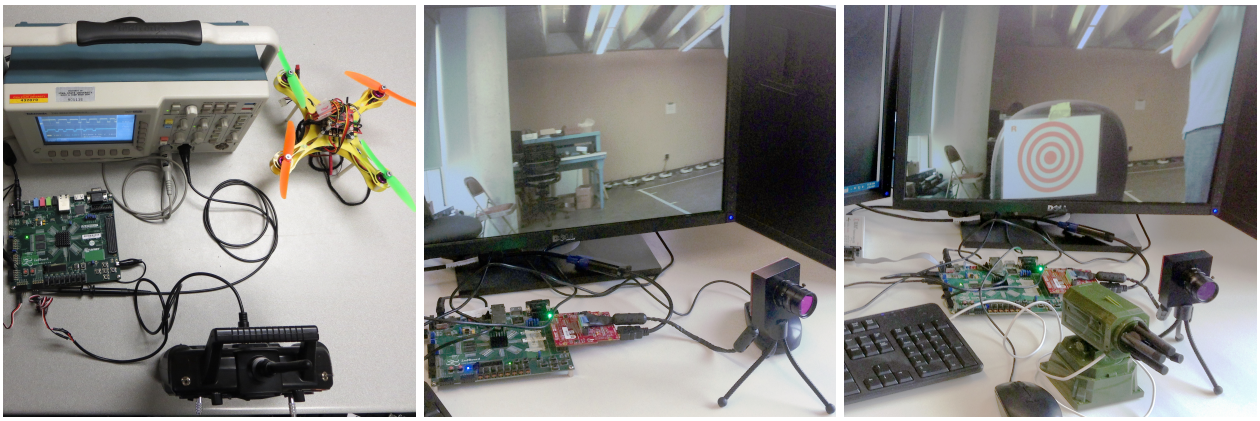
To write software for the Zynq, students use the Xilinx Software Development Kit (SDK), an Eclipse-based IDE. Software is written in C and cross-compiled for the ARM architecture using the Xilinx-distributed GCC toolchain. The SDK is also integrated with XPS to allow the low level board-support package (BSP) for a project to be exported to the SDK, providing the necessary software libraries to interact with the Zynq hardware, IP cores, and other peripherals.

IV. Laboratory Assignments

Each MP requires students to solve hardware and software problems. At the same time, each MP allows students to experience the application of concepts discussed in recent lecture modules. Students work together in groups of two or three, and are given two weeks to complete each MP. After two weeks, a portion of lecture is dedicated to video demonstrations of each group's accomplishments. Bonus points are also offered for each MP to allow more driven or advanced students to remain challenged. After the last MP, students spend the final 4 weeks working on projects they propose. The rest of the section describes each MP in detail.

A. MP-0: NES Emulator

For the first MP, students are to finish an almost complete implementation of a Nintendo Entertainment System (NES) emulator. The primary challenge for this MP is the construction of the VGA video output pipeline in hardware, using third-party IP cores from Xilinx. Once the hardware video pipeline is working, students can then add the necessary lines to the NES emulator framework to display the graphical output. Additionally, students are asked to provide support for single-player input using the buttons and switches on the ZedBoard. Bonus points are earned by adding support for a menu, menu navigation, multiple games, alternative input methods, or audio. This MP serves as a rapid introduction to the Xilinx tools that are used for the rest of the semester, as well as to the ZedBoard platform. Students are also exposed to system prototyping using memory-mapped third-party IP cores.



(a) (b) (c)

Fig. 2: Workstation setups for MP-1, MP-2, and MP-3.

B. MP-1: Quadcopter Interface

This MP introduces students to quadcopter concepts of operation, particularly, control by pulse position modulation (PPM). Students must design and implement in an HDL a memory-mapped hardware module that connects a Hobby King radio controller in trainer mode to a second controller, acting as the instructor, through the ZedBoard PMOD interface (Fig. 2a). The instructor controller is linked to a quadcopter and transmits input from the trainer controller, the software on the ZedBoard, or itself. Once the PPM-capture module is written, students must also write software that can read and write to the peripheral, and include functionality for simple pass-through of PPM frames, recording of PPM frames to memory, playback of stored PPM frames, and a filtering function to attempt to detect if a given PPM frame will put the quadcopter into an unstable position. Bonus points are earned by demonstrating autonomous flight longevity or by creating better diagnostic software on the ZedBoard and/or host workstation. This MP provides students with exposure to PPM interfaces, custom memory-mapped IP core design and integration, as well as finite state machine hardware design.

C. MP-2: Digital Camera

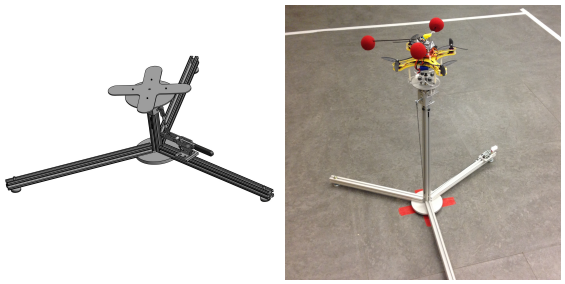
MP-2 requires students to create a digital camera using the Avnet FMC-IMAGEON FMC card coupled with an ON Semiconductor VITA grayscale image sensor. The challenge for this MP is to create a hardware video pipeline using third-party IP cores (similar to MP-0), but instead of simply drawing the camera output on a display, the final pipeline must also process the image data to convert the grayscale values to color values in the YCrCb color space (for HDMI output). However, before implementing the conversion in hardware, students must implement the conversion in a software Bayer filter, in order to compare the performance between the two implementations. The final task is to add primitive digital camera functionality: image capture, image storage, and image recall and display. Bonus points are earned by adding additional camera functionality, such as recording and playing back up to 5 seconds of video, implementing digital zoom, or adding support for manipulating other common settings, such as gain and exposure. Fig. 2b shows a working video pipeline. This MP provides students with more experience in integrating third-party IP cores, and introduces basic concepts of digital image processing and performance analysis between hardware and software implementations.

D. MP-3: Target Acquisition

MP-3 builds on MP-2 by requiring students to integrate the digital imaging system with embedded Linux to detect targets and automatically aim and fire a Dream Cheeky Thunder USB Missile Launcher (Fig. 2c). The first step is to configure and build an Open Source Linux (OSL) kernel, and construct a boot image for the ZedBoard. Students then gain experience with accessing peripherals through the operating system by writing a script that polls the switches and buttons on the ZedBoard and activates LEDs based on their current states. Next, Linux device driver development is introduced in the main challenge of this MP: extending a USB driver framework to create a device driver for the USB missile launcher. Once the driver is implemented, students add the MP-2 infrastructure to the boot image and write a Linux application to detect targets and control the USB turret. Bonus points are earned by improving the detection accuracy beyond the required 3 feet, or improving the precision of the missile launcher to reliably hit a target at a 3-foot distance. This MP focuses on embedded Linux deployment, driver development, and system programming.

E. MP-4: UAV Control

For the final MP of the course, students leverage the MP-1 infrastructure to create a proportional-integral-derivative (PID) controller to control the yaw for the quadcopter from MP-1. The PPM pass-through module is used to capture input from the Hobby King controller as before, but now the commands are sent to the quadcopter through a Bluetooth module attached to the ZedBoard PMOD interface. Additionally, the software must send throttle ARM and DISARM commands to the quadcopter, as well as request and receive telemetry from the quadcopter. To simulate GPS coordinates for the quadcopter position, the instructors developed an OpenCV application that detects three red balls attached to the quadcopter by using a camera suspended above the quadcopter pedestal (Fig. 3b) and interpreting their positions as coordinates. The application streams the coordinates to any machines running a client application. Once the software infrastructure is built, students use the data as input to their PID controller to control the yaw of the quadcopter. The goal of the PID controller is to provide stability, so when the quadcopter yaw deviates from the set point, the PID provides commands to the quadcopter



(a) (b)

Fig. 3: MP-4 quadcopter pedestal.

motors over the Bluetooth connection to return the yaw to the set point. Bonus points are earned by implementing additional controllers for pitch and roll, as well as implementing full autopilot hover or full autopilot with a short flight plan. This MP introduces students to control theory and its applications in embedded systems, as well as using a Bluetooth peripheral to collect data and interpret it in software.

1) Quadcopter Platform

In order for students to be able to test the effectiveness of their PID controllers, as well as to receive telemetry from the quadcopter, it was necessary to use a live quadcopter. However, since the lab is indoors, there are significant restrictions on flying the quadcopter, especially with in-development PID controllers. Therefore, a platform was developed specifically for this course that would secure the quadcopter, but would rotate, allowing yaw to change as the motors responded to throttle commands. A schematic and a photo of the completed device in action are shown in Fig. 3.

F. Final Project

The final project allows teams of students to investigate an in-depth topic that interests them, by either building off of a previous MP, or creating something new. Students write project proposals, consisting of a project description and rubric, midway through the semester. The proposals are presented to the class and the rubric is reviewed by the instructor to ensure it has reasonable scope to be challenging, but not so much as to cause students to struggle earning points. Any feedback from the instructor is used to revise the rubric until it is satisfactory. During the last class meeting, students present their projects and evaluate themselves against their rubric. Their classmates and the instructor also evaluate them based on their rubric. All the evaluations factor into the final project grade. Past projects have included extensions to MP-0, image processing applications, audio processing applications, signal processing applications, embedded software systems using Linux, and even emulation of other microprocessors.

V. Student Response

The overall student response for the two semesters the course has been offered has been favorable. Data collected from end-of-semester surveys revealed that the first semester the course was offered, 23 of 27 students responded to the course evaluation survey and rated the overall effectiveness of the course 4.52/5.00. The same 23 students also rated their inspiration to learn more about the course material at 4.09/5.00. The second semester the course was offered, all 19 students in the class responded, and they rated the overall effectiveness of the course at 4.68/5.00. These same students rated their inspiration to learn more about the material at 4.53/5.00.

VI. Conclusion

The course presented in this paper offers students a challenging and hands-on introduction to embedded systems design using an SoC development platform. Undergraduate students who complete this course have a unique experience that covers many of the concepts and challenges present when designing real embedded systems. We intend on continuing to offer the course in both fall and spring semesters to keep up with a growing undergraduate student population. All of the lecture and laboratory materials, and the technical documentation, for CprE 488 are publicly available on the course website [11].

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under awards CNS-1116810 and CCF-1149539.

References

[1] I. Schmädecke, C. Leibold, H.-P. Brückner, and H. Blume, "Project-organized education: From FPGA prototyping to ASIC design: Consecutive microelectronic education in designing application-specific hardware," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2013, pp. 9–12.

[2] C. Sitik, P. Nagvajara, and B. Taskin, "A microcontroller-based embedded system design course with PSoC3," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2013, pp. 28–31.

[3] A. Mondragon-Torres and J. Christman, "A comprehensive embedded systems design course and laboratory," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2013, pp. 56–59.

[4] J. Hamblen, Z. Smith, and W. Woo, "Introducing embedded systems in the first C/C++ programming class," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2013, pp. 1–4.

[5] X. Wang, "Using FPGA-based configurable processors in teaching hardware/software co-design of embedded multiprocessor systems," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2011, pp. 114–117.

[6] M. Steffen, P. Jones, and J. Zambreno, "Teaching graphics processing and architecture using a hardware prototyping approach," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2011, pp. 13–16.

[7] K.-C. Yang, Y.-T. Chang, C.-M. Wu, C.-M. Huang, and H.-H. Luo, "Application-oriented teaching of embedded systems," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2011, pp. 118–121.

[8] F. Machado, S. Borromeo, and N. Malpica, "Project based learning experience in VHDL digital electronic circuit design," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2009, pp. 49–52.

[9] J. Schneider, M. Bezdek, Z. Zhang, Z. Zhang, and D. Rover, "A platform fpga-based hardware-software undergraduate laboratory," in *Proceedings of the International Conference on Microelectronic Systems Education (MSE)*, 2005, pp. 53–54.

[10] D. Rover, R. Mercado, Z. Zhang, M. Shelley, and D. Helvick, "Reflections on teaching and learning in an advanced undergraduate course in embedded systems," *IEEE Transactions on Education*, vol. 51, no. 3, pp. 400–412, 2008.

[11] *CprE 488 - Embedded Systems Design homepage*, http://class.ece.iastate.edu/cpre488, 2015.