Creating a Realistic Embedded Systems Design Experience for Computer Engineers

Michael G. Morrow
Department of Electrical and Computer Engineering
University of Wisconsin-Madison, WI

Abstract

This paper chronicles the author's experiences in designing and implementing a capstone computer engineering design course to incorporate state of the art technology. Often, these design courses are forced to one of two extremes - one, using simpler technologies to facilitate student fabrication and testing, since modern devices in 'student friendly' packages are becoming less and less available, or two, using complex, state-of-the-art devices but at a high level of abstraction to make them accessible to students. In redesigning our embedded systems design course, we wanted to ensure that our students worked with the state of the art (i.e. current microprocessors, standard interfaces and current technology I/O devices, real-time operating systems, application and device driver software development, and hardware description languages), but that they still did actual hardware design and fabrication. We wanted students to have independence in the selection of projects, but had to ensure a uniform level of support. Finally, we had to ensure that this presented our students with a reasonably achievable design effort, and that they would have a good opportunity for success. This somewhat daunting set of goals was in fact achieved through a strategy that incorporated team design, parallel establishment of specialized 'expert teams', partnership with industry, and the establishment of a hardware/software infrastructure that helped students succeed at these new and unfamiliar tasks. The paper describes the faculty effort required in preparation for and during this course, the particulars of the implementation, and how the course has evolved over several years. The methods, mechanisms, and lessons learned that are described here may be helpful to others contemplating a similar course, or those anticipating a revision to an existing computer engineering design course.

1 Introduction

Typically, computer engineering design courses are forced to use outdated and/or simpler technologies in order to facilitate student fabrication and testing, since modern devices in 'student friendly' packages are not readily available. We made a radical shift in methodology when redesigning our embedded systems design course. This was done to expose our students to a realistic design environment. In particular, we wanted to ensure that our students worked with more modern tools and concepts while ensuring that they still did actual hardware design and fabrication. These tools and concepts include more complex state of the art microprocessors, standard interfaces and current technology I/O devices, real-time operating systems, application and device driver software development, and hardware description languages. However, we also had to ensure that this presented our students with a reasonably achievable design effort and that they would have a good opportunity for success. This somewhat daunting set of goals was in fact achieved through a strategy that incorporated team design, parallel establishment of specialized expert teams, partnership with industry, and the establishment of an infrastructure that helped students succeed at these new and unfamiliar tasks. Obviously, there is significant faculty effort required during the preparation for and in teaching this course, but these efforts can be reasonably accomplished. The methods, mechanisms, and lessons learned that are described here may be helpful to others contemplating a similar course, or to those anticipating a revision to an existing computer engineering capstone design course.

2 Course Structure and Overview

The over-arching goal of the course is to expose students to a realistic embedded system design experience, while giving them an opportunity to bring their accumulated knowledge to bear on a specific design of their choosing. The course philosophy centers on the idea that the student should work through a challenging project, but that student success with their project is a vital part of the overall experience. This course serves as a capstone design experience for students, so they have generally completed most of the curriculum by the time that they enroll in this course. The prerequisites effectively limit the course to students who are in the computer engineering program (as opposed to the electrical engineering program).

The design projects are accomplished by self-selected teams of three students. The teams are required to prepare a written contract detailing the members' responsibilities (rotation of responsibilities is required), weekly meeting arrangements, and how disputes will be resolved. The team maintains a lab notebook throughout the course, and meets weekly with the course instructor and/or teaching assistant.

The course is structured to contain both a lecture component (three 50 minute periods per week) and an intense laboratory/project component (one 3 hour period per week). The lecture component initially focuses on giving the students the required background to successfully implement a design using the course's hardware and software platforms. The focus then shifts to a sampling of more pragmatic design issues that most students have not experienced in the curriculum, including:

- Component selection, including capacitor characteristics and usage, circuit board power distribution, and basic power supply design.
- Interfacing to sensors, including a review of digital and analog signal conditioning techniques, and the characteristics and use of a variety of sensors.
- Interfacing to the telephone system, including an overview of the telephone system operation and the use of non-linear quantization.
- Low-power design techniques at various levels of abstraction.
- Interfacing techniques for inductive loads, basic motor operation, and heatsink applications and calculations.
- Advanced bus systems, typically covering the PCI bus and the evolution to current serial buses and switched fabric concepts.

Similarly, the laboratory component follows a phased approach. The initial lab meetings are structured, with all students completing the same exercises, including:

- Windows CE introduction and operating system image creation

- Windows CE applications using Microsoft Foundation Classes (MFC)
- Windows CE stream interface device driver development and usage
- FPGA-based hardware design tools and FPGA device programming methods

After the structured laboratory exercises, the laboratory time is used for weekly progress meetings with each design team and the instructor and/or teaching assistant. During this phase, the student design teams work independently on their projects.

The project teams are required to complete a number of formal deliverables as part of their project. The deliverables mirror a realistic design process, and are actually derived from the processes in place at a major design house. They serve both as an assessment tool, and to keep students' project effort more balanced over the course of the semester. The deliverables include;

1. Initial proposal - a single page narrative and block diagram of the students intended project. This is due very early, and largely serves to ensure that the project scope is reasonable. This proposal is discussed with the team, and recommendations made for changes if needed. In general, very few teams choose projects that are too easy - far more common is that they initially choose something that is probably well beyond what they should reasonably expect to accomplish in a semester.
2. Detailed proposal, work schedule, and cost/time estimate - the detailed proposal consists of an executive summary, a description of the problem to be solved, a functional block diagram and a description of the project features. Students also make a short presentation of their project to the class at this point. At this point, the teams are often directed to make certain hardware/software trade-offs. The point of this is to ensure that they have sufficient complexity in their hardware design.
3. Requirements definition - this explicitly defines the detailed functional capabilities of the device/system being built. Teams also submit an updated schedule and cost/time estimate.
4. Schematic diagram, test plan, bill of materials (BOM) - the test plan specifies in detail how the team will demonstrate conclusively that their project is in fact functional.
5. Timing analysis - teams complete an analysis of the timing margins for PC-Card bus activity at the prototyping board interface with their circuitry. Teams also submit an updated schedule and cost/time estimate.
6. Initial hardware test - teams are required to demonstrate the functionality of their hardware. They may use multiple versions of software and FPGA designs to do so. The hardware test occurs approximately 2 weeks before the final project demonstration. The primary purpose of this deliverable is to encourage teams to get their hardware working early enough to be able to complete the remainder of the project successfully. This requirement was added based on prior experience with teams often underestimating both the number of issues that will arise when they integrate their hardware components and the time that will be expended in correcting these. Since it has been in place, the results seen in the final project demonstration have improved significantly.
7. Final project demonstration - the team demonstrates the functionality of their completed project to the instructor.
8. Final project report, including a self-analysis of their project execution and lessons learned in the course.

In addition to being a member of a project design team, each student is also a member of an *expert team*. The exact function of the expert teams varies by semester, but the idea is that the

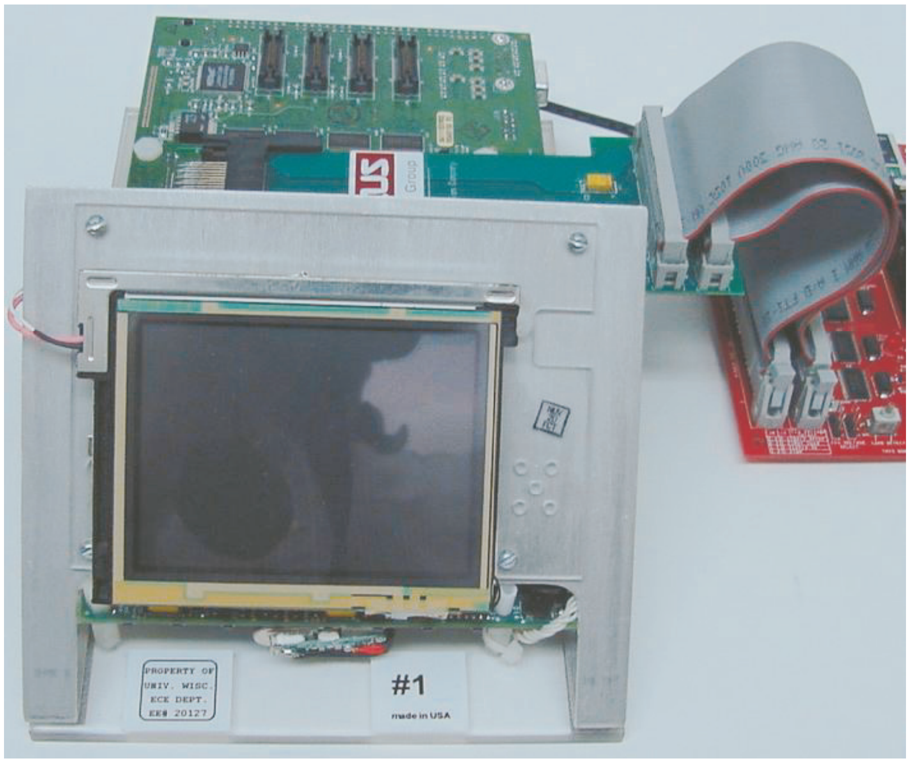

Figure 1: Photograph of the StrongARM board set (front view).

students on a given expert team will become experts on some particular area of interest, prepare a presentation, and then be available to assist other students in that area. In the past, this has been commonly used to create pockets of expertise in specific areas of interest, such as device drivers, MFC graphical elements, and the like. In addition to providing multiple sources of expertise within the class, this also mimics common industry practice where individual engineers often have an assigned (or developed) specialization in addition to their more general project work.

3 Hardware Environment

The hardware design platform used is based on the Intel StrongARM microprocessor, specifically the Assabet and Nepponset evaluation board set. This hardware provides the capabilities of a personal-digital assistant (PDA), with a touch-screen graphical display, keyboard and mouse interface, network connectivity, and numerous peripheral devices. The original evaluation board set was reconfigured and mounted on an aluminum frame to facilitate student use of the board and to protect it from handling stresses, as shown in Figures 1 and 2.

The evaluation board is in fact a complete embedded system, intended to be used by engineers to evaluate the processor and peripheral devices without having to create any prototype hardware. Given that many modern semiconductor packages are unsuitable for hand assembly, the evaluation board also serves as a generic prototype, without the expense and time of producing a unique prototype product. This is a very useful and practical design method for embedded system engineers in industry. However, it is a problem for education in that the evaluation board often leaves no room for further development by students. On the other hand, the evaluation board does make more modern and capable computer hardware accessible to education. In many typical microprocessor design courses, students will start with bare microcontrollers or minimally

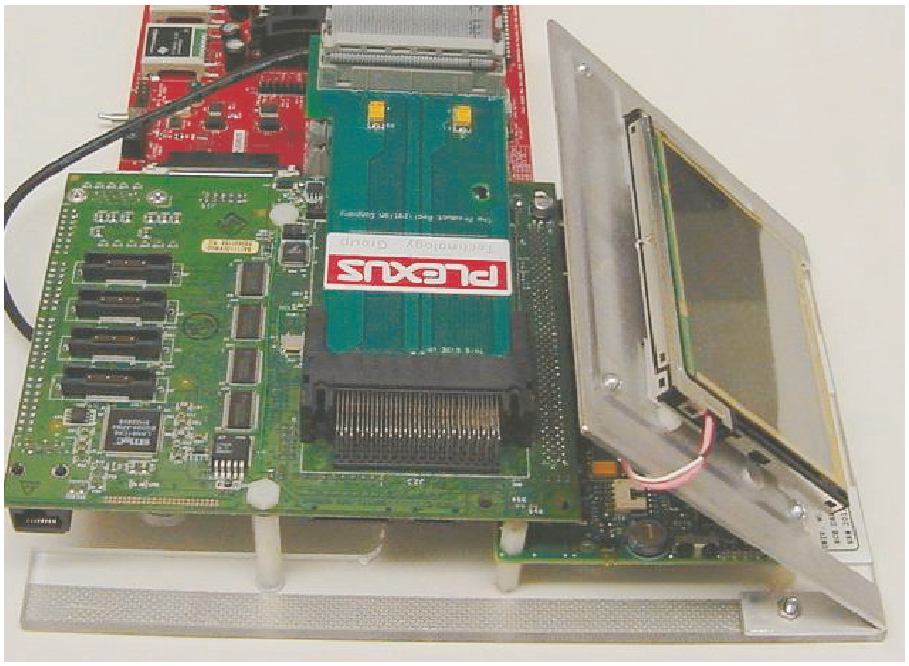

Figure 2: Photograph of the StrongARM board set (side view).

functional evaluation boards so that they are able to design further functionality into their system. The drawback to this method is that students are limited to relatively simple interfaces (both the user interface and the electrical interface), and to using parts that can be hand-assembled. Also, if students freely choose their own processor, obtaining and managing development tool licensing and support is often problematic at best. Our goal was to include state of the art hardware (such as the StrongARM microprocessor) and provide robust tool support, but still retain for students the essential elements of low-level design and fabrication.

To achieve this goal, it was decided that the prototyping target would be the PCMCIA (PC-Card) interface. The PC-Card interface is equivalent to a conventional microprocessor's parallel system bus in most respects. Thus students are exposed to a similar environment as they would be when designing for a direct interface to a microprocessor. In general, students end up designing what could be described as an intelligent peripheral device. Functionality can be shifted between the StrongARM software and the prototyping card hardware as necessary to ensure a reasonable level of hardware complexity. For example, when interfacing to a stepper motor, the prototyping board hardware could be as simple as a register that drives the motor controller inputs, with all high-level functionality implemented in software. To increase the complexity of the PC-Card hardware design, we have often required students to implement high-level control functions in hardware. In this example, control of the stepper motor would be accomplished by writing the number of steps to be taken to the PC-Card hardware, then the hardware would report completion back to the StrongARM board. This flexibility is a valuable tool in balancing the complexity of the hardware and software.

To make the PC-Card interface accessible to students, a prototyping card was developed to plug into the PC-Card card interface on the StrongARM board, as shown in Figure 3.

The prototyping card contains power control circuitry that allows powering the card either directly from the StrongARM evaluation board, or from an external power supply. The power delivery to the prototyping card is controlled by the StrongARM microprocessor in both cases, as required by the PC-Card specification. An Altera Cyclone field-programmable gate array (FPGA) on the prototyping card is available for students to use in implementing digital logic. Students typically

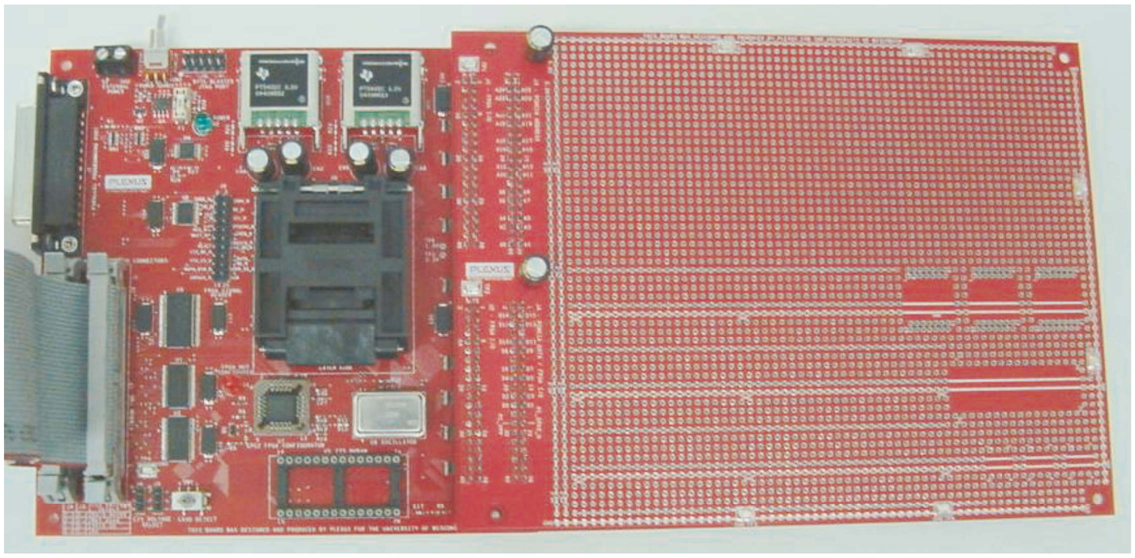

Figure 3: Photograph of the PC-Card prototyping board.

design their systems so that the majority of the PC-Card interface, state machines, and other glue logic is contained in the FPGA. The prototyping card itself is divided into two detachable sections. One section is a four-layer printed circuit board that contains the power supplies, FPGA and configuration circuit, PC-Card buffers and transceivers, and all other interface logic. This section is reused each semester. The second section is a low-cost two-layer board that is used by the students to add their custom hardware, and can be disposed of after the semester. This gives students the freedom to solder on and otherwise alter the prototyping board as they need to. A through-hole prototyping area is used to add additional hardware devices and to construct the interface to the outside world. Several sets of common surface mount pads are available in the prototyping area to allow easy use of surface mount devices. Some typical devices that students add include analog-to-digital and digital-to-analog converters, motor drivers, communications interfaces, and additional discrete logic. Students often design and build custom circuit boards as well. This enables them to effectively use a large number of surface mount devices and to minimize the hand-wiring involved.

For hardware development, various schematic capture programs and the Altera Quartus FPGA design tools are used. Most student FPGA logic design is done in Verilog, as they have been exposed to it in previous course work.

4 Software Environment

The software environment is driven by the use of the Microsoft Windows CE operating system. One of the first lab assignments students are given is to build a Windows CE operating system executable image and download it to the flash memory on the StrongARM evaluation board. They are introduced to Windows CE in several classroom lectures, covering its basic structure, operation, and the development tool flow. In the lab, they learn how to create a basic Windows CE image. No significant attempt at operating system customization is made beyond the basics of incorporating registry settings and additional files into the operating system image. This somewhat cursory approach is quite acceptable since the focus of the class is embedded system design and not operating systems.

Students are required to create a stream interface device driver using Embedded Visual C++ in order to access the PC-Card prototyping board address spaces from a Windows CE application.

Again, the concepts of dynamic link libraries (DLLs) and device driver loading are introduced in lecture. This is followed by a structured laboratory exercise that walks students through the creation of a skeleton stream interface driver starting from a basic DLL.

All application development is done using Embedded Visual C++. Students are required to use the device driver that they wrote for all communication between their application(s) and their hardware. Many students have minimal exposure to C/C++ programming, however, they quickly adapt to the use of the Microsoft Foundation Classes (MFC) for their graphical interface. The basics of MFC application development and the use of the remote debugging features of Embedded Visual C++ are explored in another structured laboratory exercise.

5 Faculty Involvement

To make a course like this work requires faculty involvement in two ways. First, the initial establishment of a course like this is quite involved. It requires the procurement of hardware and software tools and the time to become intimately familiar with them. Once the course is in place, there must be a continuing effort to maintain the hardware and software, and to deal with obsolescence and upgrades. The lecture component should be continuously modernized to remain relevant. Fortunately, this is quite a bit less work than the initial effort. The key here is the continuity of support by faculty - this is not a responsibility that can be passed from person to person and still have a reasonable expectation that the course will not be degraded.

The course is taught with a teaching assistant, but the primary role of the teaching assistant is to manage the progress of design teams. Of course, they also serve as a technical expert as their experience permits. Realistically, the project teams encounter so many varied design issues that the faculty member needs to remain constantly involved.

6 Conclusions

This course has proven to be quite popular with students, routinely receiving very positive feedback on student evaluations. A more satisfying result has been positive feedback from students (and interviewers) about how they were able to use the course experience in job interviews. Although the course requires a strong faculty commitment, it provides an excellent capstone design experience for our students.

The web page for this course [1] is publicly available for information and reference. Copies of the current course syllabus and pictures of prior semester projects are among the items available. Interested parties are also invited to contact the author via e-mail.

7 Acknowledgements

The author wishes to express sincere gratitude for the support received from Intel Corporation, Microsoft Corporation, the Plexus Technology Group, and PCMCIA. Their generous contributions of hardware, software, and design support have been a critical element in the successful development and continuation of this course.

References

[1] M. G. Morrow, "University of Wisconsin at Madison," 2004. `http://eceserv0.ece.wisc.edu/~morrow/software/`.

MICHAEL G. MORROW, P.E., is a Faculty Associate in the Department of Electrical and Computer Engineering at the University of Wisconsin, Madison, WI. His research interests include real-time digital systems, embedded system design, software engineering, curriculum design, and educational assessment techniques. He is a member of ASEE and IEEE. E-mail: `morrow@ieee.org`