

Developing and Teaching an Integrated Series of Courses in Embedded Computer Systems

Mikel Bezdek¹, Daniel Helvick², Ramon Mercado³, Diane Rover⁴, Akhilesh Tyagi⁵, Zhao Zhang⁶

Abstract – With embedded computer systems being a core topic in computer engineering, there are typically one or more courses in a program that provide varying coverage. Many universities offer introductory courses that focus on microcontroller-based systems and embedded programming. Advanced courses often do not have a common focus and are not available until the graduate level, leaving a gap in training undergraduates. At Iowa State University, the Department of Electrical and Computer Engineering developed a new senior-level design course on embedded systems design (CPRE 488) that bridges the content between the introductory course on microcontrollers (CPRE 211) and a graduate course on system-level design (CPRE 588). This paper presents the process of developing the integrated series of courses that spans early undergraduate to graduate levels, including the team approach used. The set of courses and the development process should be of interest to educators considering expanding or enhancing the curriculum in embedded systems.

Index Terms – Embedded systems, Hardware-software co-design, Problem-based learning, System design methodology

INTRODUCTION

The need addressed by the curriculum development described in this paper is expressed in the following statement by Bordogna [1]: "Most curricula require students to learn in unconnected pieces - separate courses whose relationship to each other and to the engineering process are not explained until late in a baccalaureate education, if ever. Further, an engineering education is usually described in terms of a curriculum designed to present to students the set of topics engineers "need to know," leading to the conclusion that an engineering education is a collection of courses. The content of the courses may be valuable, but this view of engineering education appears to ignore the need for connections and for integration - which should be at the core of an engineering education."

We have developed a series of courses spanning several years, from introductory to advanced, to engage students in different perspectives on the design of embedded computer

systems. The courses have overlapping and complementary content. The first course introduces students to embedded system components and embedded programming using both bottom-up and top-down techniques. The second course emphasizes integration of components into a system implementation using advanced tools that support bottom-up design. The third course focuses on high-level abstraction and top-down, system-level design methodologies that start with a specification model of the system. The courses are integrated through a coordinated set of learning outcomes and the use of related tools and technologies. In addition, the courses are designed with special attention to integrating the lecture and laboratory experiences, making explicit the relationships between lecture topics and laboratory exercises.

In this paper, we first introduce the courses. Then we present the pedagogical approaches using a design case study. We conclude with observations on the courses and comparisons to embedded systems education.

COURSE OVERVIEW

I. CPRE 211: Microcontrollers and Digital Systems Design

CPRE 211 is a sophomore-level course. It was revitalized in 2001 by introducing an MPC555-based platform called PowerBox [2]. The goal was to develop an interesting, integrated classroom/laboratory experience for the students. Meanwhile, the use of PowerBox and the associated CodeWarrior were representative of technology at the time in embedded systems design. To achieve the goal, problem-based learning is emphasized in the lab exercises. Most labs are designed to resemble real-world applications in precise agriculture. For example, track meters and sprayer locks. There are approximately ten lab exercises performed in groups of two or three students and a lab project performed in larger groups of students. Students exercise good programming style, modular design, debugging skills and teamwork through the semester. Most labs have a pre-lab and SKIBLE (SKILL Building Exercise) to get students involved more deeply.

The course is also designed to fully cover the subject of microcontroller-based systems design, including embedded hardware models, embedded programming in C, simple I/O interfaces, assembly programming, mixed C/assembly

¹ Mikel Bezdek, Graduate Student, Computer Engineering, Iowa State University, mbezdek@iastate.edu

² Daniel Helvick, Graduate Student, Computer Engineering, Iowa State University, dhelvick@iastate.edu

³ Ramon Mercado, Graduate Student, Computer Engineering, Iowa State University, rmercado@iastate.edu

⁴ Diane Rover, Associate Dean, College of Engineering, Iowa State University, drover@iastate.edu

⁵ Akhilesh Tyagi, Assistant Professor, Department of Electrical and Computer Engineering, Iowa State University, tyagi@iastate.edu

⁶ Zhao Zhang, Assistant Professor, Department of Electrical and Computer Engineering, Iowa State University, zzhang@iastate.edu

programming, interrupt-based design, and programming advanced I/O devices. Those knowledge areas are covered by lecture notes and carefully integrated into the lab exercises.

II. CPRE 488: Embedded Systems Design

CPRE 488 is a senior-level course, wherein the goal is to develop system-level design experience for the students. It was created in fall 2005 to bridge the gap between CPRE 211 and CPRE 588, and to reflect recent technology advancements. Students have studied computer organization and operating systems, and optionally software engineering and real-time operating systems. This course introduces and reviews hardware and software design issues from a system-level perspective, including hardware/software interfacing, compiler techniques, profiling methods, hardware accelerators, testing methods, real-time scheduling, multiprocessor-based designs, and networked systems. It also gives students rich opportunities to exercise software engineering methods, including system design processes and UML (Uniform Design Language) methods. The lectures are based on Wolf's [3] book.

As in CPRE 211, the labs are a significant component and are integrated with classroom teaching. The lab platform hardware consists of Xilinx Virtex II Pro boards from Digilent shown in figure 1. The boards have a Xilinx XC2VP7 FPGA chip with 30,816 Logic Cells, 136 18-bit multipliers, 2,448Kb of block RAM, two PowerPC 405 processor cores, and DRAM support of up to 2GB. They also have rich I/O capabilities including Ethernet, USB, Video/audio ports and gigabit serial ports, among others. This platform is very close to the industry standard. It exposes the students to the rich features and full complexity of contemporary embedded systems design toolsets. The design environment for the labs is the Xilinx EDK/ISE development environment, as well as VxWorks RTOS to support real time programming.

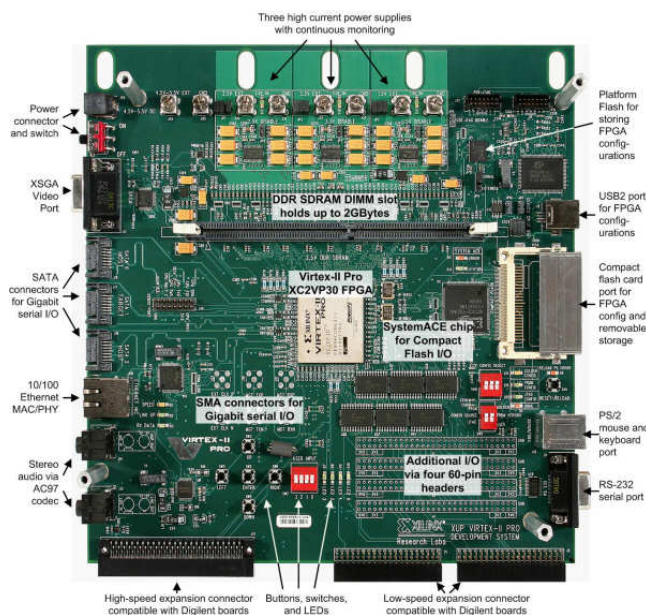


FIGURE 1
XILINX VIRTEx-II DEVELOPMENT BOARD FROM DIGILENT

Most labs are designed for problem-based learning and are based on two real-world applications, digital cameras and MP3 players. They are interesting applications. They are also good systems to use for demonstrating the concepts of performance analysis, profiling methods, hardware accelerator design, real-time scheduling, and system testing. There is an open-ended capstone project, in which students develop new applications based on their lab experiences. Students worked in groups of two in the lab exercises and in a team of two or three groups in the project. Because of the carefully designed lab exercises, all students were able to implement some working and impressive systems, including feature-enhanced MP3 players, an Internet Radio player, a miniature recording studio using the on-board AC97 audio codec, and a 3D rendering engine.

III. CPRE 588: Embedded Computer Systems

CPRE 588 is a graduate-level course focusing primarily on design methodologies and modeling languages for embedded computer systems, as well as various models of computation. Graduate students, as well as qualified undergraduate students, are allowed to take the course. CPRE 588 is also offered as a distance education course, so the background of the students of CPRE 588 is much more diverse than that of either CPRE 211 or CPRE 488. Typically, students of CPRE 588 have some background in operating systems and real-time concepts, as well as networking, algorithm design, and some kind of previous embedded systems design experience.

The embedded systems design methodology taught in CPRE 588 is discussed at length in [4], the textbook for the course. The methodology involves starting with a high-level, abstract, functional description of an embedded system and gradually refining it to be more concrete. SpecC is a modeling language that accompanies the methodology presented in [4]. It is a C-like language that presents many new constructs that are of use to embedded system design, such as communication channels, bit vectors, state machines, events and transactions, and timing constraints. The refinement models in the SpecC methodology are:

- **Specification Model** – High-level, abstract model. No implicit structure or architecture. Un-timed execution.
- **Architecture Model** – Component structure and architecture. Behaviors grouped under top-level component behaviors. Sequential behavior execution. Timed model with estimated execution delays.
- **Communication Model** – Component and communication bus structure and architecture. Timing-accurate bus protocols. Timed model with estimated component delays.
- **Implementation Model** – Cycle-accurate system description. Object code for processors. Clocked bus communications.

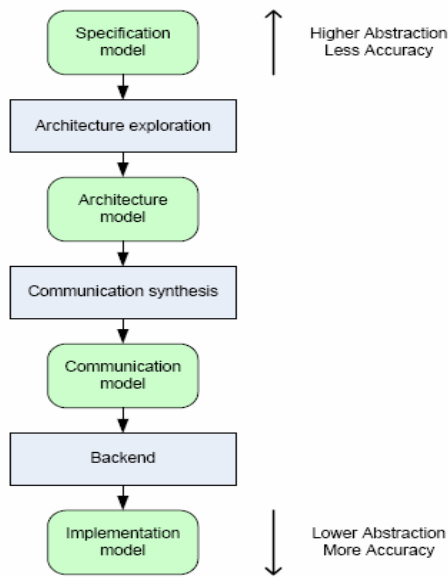


FIGURE 2
SPEC C DESIGN METHODOLOGY MODELS (FROM [6])

A number of refinements are made to each model in order to progress to the next model. For instance, to progress from the specification model to the architecture model, high-level behaviors are partitioned onto distinct processing elements in the system, and abstract communication channels are placed to communicate between the processing elements in the architecture exploration phase. To progress from the architecture model to the communication model, specific protocols are selected for the communications channels and inlined into the design in the communication synthesis stage. The backend stage synthesizes the design to the implementation level automatically through the use of tools.

SystemC is a library of functions for C++ that are similar in function to SpecC. SystemC is also taught in CPRE 588, but the emphasis remains on SpecC and the SpecC design methodology.

There are **no scheduled laboratory** exercises in CPRE 588. However, there is a **capstone design project** completed in teams of approximately four students. The students choose an embedded system (for instance, an mp3 player), and go through the SpecC design methodology using the system they choose. The students are asked to make a series of refinements to their models in order to approach the implementation-level model.

CASE STUDY IN PEDAGOGY

The pedagogical approaches used to teach system design in CPRE 488 are fundamentally different from those used in CPRE 588. In CPRE 588, a high-level, top-down approach to system design is presented. Design begins by specifying the functional behavior of the system, and then follows iterative refinements leading to an actual implementation. On the other hand, to bridge the gap with introductory courses, CPRE 488 presents a bottom-up approach to system design. First, components are designed and analyzed, and then the system is

built from these components. This approach gives undergraduate students a practical understanding of the issues surrounding system design. To highlight the differences between the courses, we will first present a common example, a digital camera, and then show how it is taught in both CPRE 488 and CPRE 588.

The digital camera example used is based on a simplified system model described by Vahid and Givargis in [5]. The camera system has only one function, to capture, process, and store images. A state diagram of the system is shown in figure 3. The task of capturing a picture starts when the user presses the shutter button. At this time, a digital image is captured by a CCD (charged-couple device). The raw pixel data is sent from the CCD to the encoding process, which begins immediately after capture. During encoding, the captured image is first transformed into the frequency domain using a two dimensional forward discrete cosine transform (2-D FDCT), then quantized and encoded using Huffman encoding. Finally, the resultant JPEG image is stored into memory. This example was chosen as it typifies the tasks that many traditional embedded systems perform: obtaining data from the environment and processing that data into a usable format. The following sections show how this system evolves in the differing course design approaches.

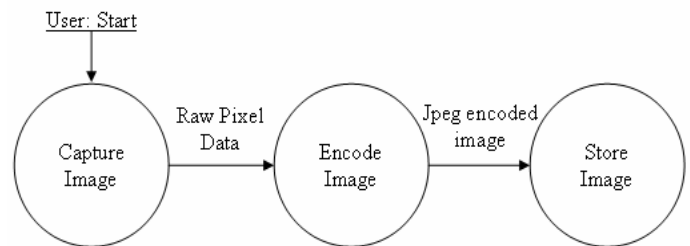


FIGURE 3
STATE DIAGRAM OF THE SIMPLIFIED DIGITAL CAMERA EXAMPLE.

I. CPRE 488

The digital camera example is used to achieve several of the learning objectives in CPRE 488. The specific learning objectives for CPRE 488 that are addressed with the digital camera exercises are as follows:

- Gain an understanding of the working principles of embedded systems and their components
- Learn how to integrate embedded hardware and software to meet the functional requirements of embedded applications
- Gain an understanding of basic performance analysis

The first learning objective stated above is achieved through the **introduction** of the use of custom hardware components and how they are integrated. In the case of the digital camera exercises in this course, the custom hardware component is the camera component and its device driver. The Xilinx Virtex II Pro platform used in these exercises introduces the students to additional tools and components and how to efficiently use them in a cohesive design process. The

second learning objective is achieved by leading the students through a structured design process that interchanges the use of software and hardware components in the digital camera to meet timing requirements of the processing of the image taken by the camera. The third learning objective is achieved by using profiling tools to determine the areas of execution that require the most improvement to achieve the required level of performance in the camera's image processing.

In the CPRE 488 lab sequence, students start with an all-software implementation of the digital camera controller and a set of QoS constraints to meet. Students then traverse the design space from the all-software implementation that was given, to a mixed hardware/software implementation in order to meet the given QoS constraints. This exercise was carefully designed to keep students inside a fixed design flow, with the purpose of showing how different design and debugging techniques, such as profiling and hardware acceleration, are used in a realistic design.

The first step in this design space exploration is to profile the all-software implementation and find the critical functions in the design. Our design environment, Xilinx EDK/ISE, allows for software profiling with GNU tools. The students are asked to identify the functions in the software implementation that are taking the majority of the execution time in the process of encoding the captured image. A sample of the profiling data that the students collect is shown in figure 4. As can be seen, the profiling application provides information such as the number of calls to a function as well as the time spent in that function. This data is then used to guide the design refinements.

Flat profile:

Each sample counts as 0.0001 seconds.

time	% cumulative	seconds	self seconds	calls	self s/call	total s/call	name
21.82	6.38	6.38		53	0.12	0.12	get_frame
10.74	9.52	3.14					__floatsisf
8.31	14.70	2.43					__pack_d
5.07	17.90	1.48					__unpack_d
4.67	19.26	1.37					__pack_f
3.60	22.57	1.05					__unpack_f
2.17	23.20	0.64					fixdfsi
2.02	24.41	0.59		9504	0.00	0.00	JpegEncoder

FIGURE 4
SAMPLE PROFILING OUTPUT

After the profiling exercise, the students are asked to apply some software techniques to improve the design and come closer to meet the QoS constraints. While this step is necessary to show the student the software techniques that are available for improving computation time, the QoS constraints in these exercises have been carefully chosen so that these software improvements will not result in meeting the QoS constraints. The students are then asked to identify which functions are candidates for implementation in hardware. This decision is made based on profiling data gathered after implementing software optimizations for the image encoding.

In the final exercise involving the digital camera, students are asked to integrate a custom hardware component into the digital camera system. This component is provided to the students, and they are asked to interface it with their digital

camera systems. In this process, the students are able to achieve the QoS constraints for the image processing time. The students are then asked to profile this new system and perform a comparative analysis of the three digital camera systems implemented in these exercises (software-only, software-only with optimizations, and software-hardware hybrid).

II. CPRE 588

In CPRE 588, the digital camera example is used to achieve the following learning objectives:

- System-level design of embedded systems comprised of both hardware and software
- Investigate topics ranging from system modeling to hardware-software implementation

The students in CPRE 588 are first introduced to the specifications of the digital camera and a functional model, which is implemented in SpecC. Throughout the course, this initial functional model is refined to reflect the different steps in the design flow of an embedded system comprised of hardware and software components. Students follow the design of the digital camera system from specification model to implementation model, observing and performing many of the refinements in this process.

Modeling an embedded system, such as the digital camera, with SpecC is very useful during the subsequent refinement steps. Figure 5 shows how the digital camera system looks through the eyes of a SpecC communication model. In this model, some details about the communication have been decided, as well as the partitioning of tasks to processing elements. However, this system is still at a more abstract level than the implementation model of the system used in 488.

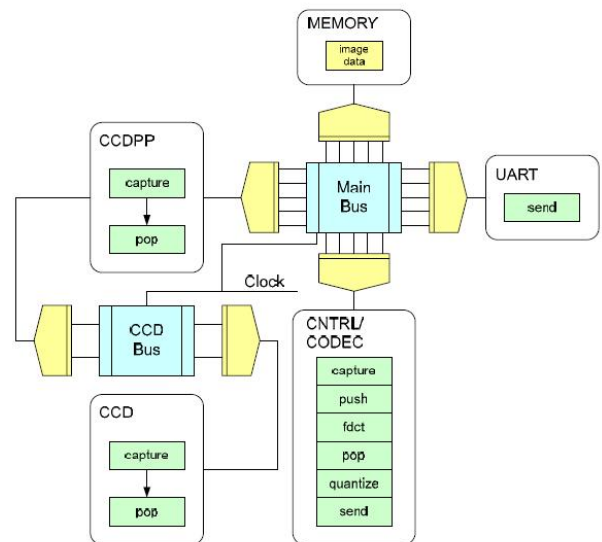


FIGURE 5
SPEC C COMMUNICATION MODEL OF THE DIGITAL CAMERA SYSTEM (FROM [6])

In [6], it is shown how the same SpecC refinement methodology may be used when designing an embedded system with SystemC. The system used in is the digital camera example used in this case study. The modeling of the digital camera in SystemC is examined in CPRE 588, and comparisons are drawn between the SpecC and SystemC modeling languages. Through this experience, students gain a greater appreciation for embedded systems design from a top-down perspective.

OBSERVATIONS ON COURSE DESIGN AND DELIVERY

In CPRE 211, the lab exercises and lectures are carefully synchronized so that the lecture part and the lab part of a given topic are close to each other. This arrangement is particularly important for students at the sophomore level. Most of them have never seen bitwise operations, assembly code, interrupt systems, ADC programming, and the like before this course. It is best for them to comprehend the intricate concepts in those subjects with both classroom and hand-on experiences.

A schedule was created in which the lecture and lab parts on important topics are at most one week apart. This was not trivial. It required fine-tuning the timing of the lectures and labs. The instructor usually starts the first class in each week with an introduction to the lab in that week, and makes conceptual connections to the related topics in the lectures. The lecture notes also use code examples from the lab exercises to make direct connections. Overall, students found the lab experience interesting and rewarding, while the lectures were able to discuss some intricate concepts in depth.

In CPRE 488, we found that here too it was challenging to synchronize the labs and lectures. We believe the issue is common to embedded system courses that use contemporary toolsets and conventional textbooks. The Xilinx EDK/ISE development environment has two interdependent sets of tools, one for FPGA hardware design and the other one for system software development. To configure and use such an environment, students must have full-spectrum knowledge in hardware and software. The lectures are based on Wolf's book [3], which uses a bottom-up organization. Hardware topics, such as CPU, memory, I/O and component interfacing are introduced before software topics such as profiling, performance analysis and real-time scheduling. This inconsistency makes it difficult to maintain the connection between the lectures and the labs. If this issue is not addressed, students may find labs less informative or lectures less interesting.

The teaching staff reorganized the sequence of lectures such that the basics of some topics are introduced in the early portion of the semester. Within the first six weeks, students had exposure to embedded CPUs and memory components, common I/O devices, FPGA basics, hardware accelerators, and compiler techniques. This reorganization helped prepare students for the lab exercises. Although the sequences of some topics were broken into two parts, we found that the benefit of early full-spectrum exposure more than offset the disadvantage. Students were able to maintain a big picture of

the systems and understood each topic from a system perspective.

We used an instructional team familiar with the introductory and advanced courses to develop CPRE 488. The team consisted of several faculty responsible for teaching the courses, as well as graduate and undergraduate students who had done one or more of the following: served as a teaching assistant in the CPRE 211 laboratory, taken the CPRE 588 class, and/or mentored senior design projects on the CPRE 488 lab platform. The team used a structured approach to organize the course and laboratory, via a learning model called 3C5I that incorporates both Bloom's taxonomy and problem based learning. The 3C5I model creates an educational context based on Concepts within Courses within a Curriculum (3C), and in each, progressing along the five "I's" of Introduction, Illustration, Instruction, Investigation, and Implementation.

Problem-based learning may extend through to either Investigation or Implementation, and in each case, to differing degrees depending on the scope of the problem. In developing the embedded systems course, we considered each learning outcome for the course and set targets for the level of learning to be accomplished in lecture versus laboratory. That approach guided the timing of presenting topics and also the depth of the hands-on exercises in the laboratory. In some cases, the purpose of a lab exercise is merely to illustrate a concept introduced in lecture; in other cases, to reinforce the instruction given in lecture; and in others, to let the student investigate independently through more difficult lab work. Using a learning model as the foundation facilitated the collaboration of the instructional team. The quality of the course and laboratory was enhanced due to the team effort.

RELATED WORK

Berkeley's embedded system design education program [7] consists of undergraduate and graduate coursework. New advanced graduate courses in many areas are under development to support research results. One sophomore course in the area exists. Efforts are currently under way to develop a junior/senior level course in mixed hardware-software systems design, similar to CPRE 488. There are also plans to develop courses based on the theoretical foundations of embedded systems design.

Carnegie Mellon [8] differentiates between the many application areas of embedded computer systems, and aims to give a broad, if not complete, exposure to these areas within its undergraduate curriculum. Many of the courses are taught as capstone design courses. Some areas of emphasis include control systems, signal processing, systems-on-chip, networking, critical systems, robotics, and security.

Embedded systems education at Vanderbilt [9] takes a model-based approach. Due to faculty size constraints, a minimal number of technical elective courses were added to introduce a specialization in embedded systems design. There are efforts underway to integrate and revise courses in systems

science and computer science to better accommodate embedded systems education.

An embedded systems curriculum is presented in [10]. It identifies sixteen core educational areas essential to such a curriculum, with emphasis being placed on programming fundamentals, digital logic design, computer architecture, software engineering, systems performance, and embedded systems design.

A survey work by the European Artist Education Group, [11], identifies several challenges for developing a curriculum in embedded systems, as well as key bodies of knowledge that should be included. The group emphasizes the need for lab experiences to strengthen understanding of core themes, which has been one of the forefronts of our course development. However, the bodies of knowledge are targeted for an entire graduate curriculum and are broad in scope. Our work is a stream of embedded courses from undergraduate to graduate level and thus is bounded by what can be covered in a few semesters' time.

Lastly, a didactic analysis of embedded systems education [12] suggests that the subject will be best taught through functional examples in an interactive setting. The lab sequence for CPRE 488 provides two such in depth examples that allow students to gain a deep familiarity with the material.

FUTURE WORK

The department is considering two changes to the existing embedded course sequence. First, CPRE 211 may be revised again to reflect recent technology advances and application changes. Its strength in problem-based learning will be kept, and new applications such as robot control and sensor systems may be introduced. Also, a new course is being considered to fill the gap between CPRE 211 and CPRE 488. It will be focused on embedded applications of intermediate complexity such as DSP systems, handheld computers, and network processor-based systems. CPRE 211 will remain an introductory course, and CPRE 488 will be more focused on the hands-on experience of hardware/software co-design.

CONCLUSION

To effectively prepare embedded systems engineers at Iowa State University, a series of courses have been developed. The courses are designed to provide an interactive, problem-based experience that encourages a deep understanding of system design issues and methodologies. The introductory course material focuses on building an understanding of basic but necessary concepts such as programming microcontrollers and designing applications to use interrupts. As students progress through the course sequence, the next course continues to build on the knowledge gained by teaching system design from a bottom-up approach. Finally, when the graduate level is reached, students are taught to design systems from the top-down.

As seen in the related work, embedded design is an exciting field for educators, albeit one that is difficult to properly address as it encompasses many existing engineering disciplines. In developing this course sequence, we have been

able to tie the myriad of concepts together into a cohesive learning experience. There is still work to be to determine what concepts need more or less emphasis and to continue to provide the functional skills that will keep graduates current with the ever-changing face of embedded systems.

ACKNOWLEDGEMENTS

This work was partially supported under NSF grant no. EEC-0088071, and through support from Rockwell Collins Foundation and Xilinx, Inc.

We gratefully acknowledge the contributions of Andrew Larson, Jason Boyd, Joe Schneider, Robert Walstrom, and Jeff Parent to the development of the CPRE 488 laboratory.

REFERENCES

- [1] J. Bordogna, "Next generation Engineering: Innovation Through Integration," Keynote Address, NSF Engineering Education Innovator's Conference, April 8, 1997, www.nsf.gov/od/lpa/forum/bordogna/jb-eaic.htm
- [2] A. Striegel, D. Rover, "Evolution of an Introduction to Embedded Systems Laboratory," *Frontiers in Education (FIE)*, Nov. 2002.
- [3] Wayne Wolf, "Computer as Components: Principles of Embedded Computing System Design," Morgan Kaufmann Publishers, 2001.
- [4] Daniel Gajski, Jianwen Zhu, Rainer Domer, Andreas Gerstlauer, Suging Zhao, "SpecC: Specification Language and Methodology", Springer, 2000.
- [5] F. Vahid and T. Givargis, "Embedded System Design: A Unified Hardware/Software Introduction", John Wiley and Sons, 2002.
- [6] Robert Walstrom, "System Level Design Refinement Using SystemC", Iowa State University, 2004
- [7] Sangiovanni-Vincentelli, A. L. and Pinto, A. "An overview of embedded system design education at Berkeley." *Trans. on Embedded Computing Sys.* 4, 3 (Aug. 2005), 472-499.
- [8] Koopman, P. et al. "Undergraduate embedded system education at Carnegie Mellon." *Trans. on Embedded Computing Sys.* 4, 3 (Aug. 2005), 500-528.
- [9] Sztipanovits, J. et al. "Introducing embedded software and systems education and advanced learning technology in an engineering curriculum." *Trans. on Embedded Computing Sys.* 4, 3 (Aug. 2005), 549-568.
- [10] Seviora, R. E. "A curriculum for embedded system engineering." *Trans. on Embedded Computing Sys.* 4, 3 (Aug. 2005), 569-586.
- [11] Caspi, P. et al. "Guidelines for a graduate curriculum on embedded software and systems." *Trans. on Embedded Computing Sys.* 4, 3 (Aug. 2005), 587-611.
- [12] Grimheden, M. and Törngren, M. 2005. "What is embedded systems and how should it be taught? -results from a didactic analysis." *Trans. on Embedded Computing Sys.* 4, 3 (Aug. 2005), 633-651.
- [13] CPRE 211 class URL: <http://class.ece.iastate.edu/cpre211/>
- [14] CPRE 488 class URL: <http://class.ece.iastate.edu/cpre488/>
- [15] CPRE 588 class URL: <http://class.ece.iastate.edu/cpre588/>