

# Reflections on Teaching and Learning in an Advanced Undergraduate Course in Embedded Systems

Diane T. Rover, *Senior Member, IEEE*, Ramon A. Mercado, *Student Member, IEEE*, Zhao Zhang, *Member, IEEE*, Mack C. Shelley, and Daniel S. Helvick, *Member, IEEE*

**Abstract**—An integrated series of courses on embedded systems has been developed at Iowa State University, Ames, spanning early undergraduate to graduate levels. The newest course in the series is CPRE 488: Embedded Systems Design, an advanced undergraduate course that fills a gap in the curriculum by providing system-level design experiences and incorporating new technology advancements. CPRE 488 development focused on lecture–lab integration and laboratory learning. Course and lab activities were designed using a learning model that captures lower-order and higher-order cognition levels of Bloom’s taxonomy. The learning experience in the laboratory is characterized using a technique to assess cognitive behavior. Results of applying the Florida Taxonomy of Cognitive Behavior are presented to summarize the depth of student learning and the opportunities for students to progress to higher-order thinking in the laboratory. After two years of experience with the new course, the authors reflect on the course design and outcomes, from both disciplinary and pedagogical viewpoints.

**Index Terms**—Cognitive behavior, curriculum integration, embedded computer systems.

## I. INTRODUCTION

AS computer-based systems take on many forms and functions in everyday life, the subject of embedded computer systems has become a core topic in computer engineering, and has also become increasingly relevant in systems design throughout engineering. The range of technologies is constantly broadening and diversifying, and curricula and laboratories are challenged to keep pace. There are typically one or more courses in a program that provide varying coverage of embedded systems. 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, Ames, 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 an introductory course on microcontrollers (CPRE 211) and a graduate course on system-level design (CPRE 588). The series of courses engages students in different perspectives on the design of embedded computer systems. 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.

Despite the fundamental role of laboratories in engineering education, the emphasis has been on laboratory development and teaching [1], rather than on understanding and assessing the laboratory learning experience. In 2002, the Accreditation Board for Engineering and Technology (ABET) held a colloquy to explore the issue of laboratory learning, resulting in a list of 13 objectives that define the fundamental purpose of educational laboratories in engineering [2]. Felder discussed critical skills development in the engineering laboratory, and Smith talked about inquiry and cooperative learning in the laboratory. A panel later revisited the objectives with a further look at assessment [3]. One of the few published efforts in understanding lab learning was written by Newstetter and Turns in 2003, relating the laboratory learning experience to the problem-based learning experience in the classroom [4]. Indeed, there is a gap between classroom and laboratory learning, stated in the concerns raised by Graham over 20 years ago, as noted by Peterson and Feisel (excerpted from [5]):

“The need for a better understanding of the teaching/learning process in the laboratory is evident. There appears to be little relevant research in this area.... This lack of research may be because nothing unique happens in the laboratory. If this is the case, we have adopted an expensive alternate mode of instruction. A more probable situation is that we have been working on the wrong problem, concentrating on “what” (goals, specific experiments, etc.), “how” (equipment setup, data acquisition, etc.), rather than “why” (an understanding of learning through the experience). In that case, we are guilty of the engineer’s greatest error—leaping to problem solution without understanding the problem” (cited as [6] by [5]).

CPRE 488 was developed with an emphasis on laboratory learning at higher-order cognitive levels. The authors have assessed cognitive learning in the laboratory to gain a better understanding of the laboratory learning experience and put this in

Manuscript received July 10, 2007; revised December 20, 2007. Published August 6, 2008 (projected). This work was supported in part by NSF Grant 0431924, by a GAANN Grant from the U.S. Department of Education to the Information Infrastructure Institute at Iowa State University, by the Rockwell Collins Foundation, and by Xilinx.

D. T. Rover, R. A. Mercado, and Z. Zhang are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: drover@iastate.edu; rmercado@iastate.edu; zzhang@iastate.edu).

M. C. Shelley is with the Department of Statistics and Political Science, Iowa State University, Ames, IA 50011 USA (e-mail: mshelley@iastate.edu).

D. S. Helvick is with Garmin International, Inc., Olathe, KS 66062-3426 USA (e-mail: dhelvick@iastate.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TE.2008.921792

the context of learning in the course as a whole. CPRE 488 fills a need relative to system-level design of embedded systems, and benefits students whether they are pursuing an industrial career or graduate studies.

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 courses and the development process should be of interest to educators considering expanding or enhancing the curriculum in embedded systems. The advanced undergraduate course is described in detail, with a focus on lecture–lab integration and laboratory learning. Results of applying the Florida Taxonomy of Cognitive Behavior (FTCB) [7] are presented to summarize the depth of student learning supported in the course. From their two years of experience with the new course, the course developers reflect on the course design and outcomes. The paper concludes with comparisons to embedded systems education.

## II. COURSE OVERVIEW

Three courses are highlighted as central to the embedded systems curriculum at Iowa State, and CPRE 488 is the centerpiece of the series [8], [9].

### A. Embedded Systems Curriculum

The first course in the series, CPRE 211: Microcontrollers and Digital Systems Design, introduces students to embedded system components and embedded programming, using both bottom-up and top-down techniques [10]. The second course, CPRE 488: Embedded Systems Design, emphasizes integration of components into a system implementation, using advanced tools that support bottom-up design and aspects of hardware/software (HW/SW) codesign [11]. The third course, CPRE 588: Embedded Computer Systems, focuses on high-level abstraction and top-down, system-level design methodologies that start with a specification model of the system [12].

CPRE 211 is a sophomore-level course organized as three lecture hours and a two-hour lab session weekly. This course was revitalized in 2001 by introducing an MPC555-based platform called the PowerBox, shown in Fig. 1 [13], [14]. The goal was to develop an interesting, integrated classroom/laboratory experience for the students. The use of the PowerBox and the CodeWarrior integrated development environment were representative of the then-current technology 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 a specific domain, e.g., precision agriculture. Students develop good programming style, modular design practices, debugging skills and teamwork throughout the semester. Labs include prelab and skill-building exercises to get students involved more deeply. The course covers an array of topics under microcontroller-based systems design, including hardware models, embedded programming in C and assembly, I/O interfaces and programming, and interrupt systems. Topics are covered in lectures, accompanied by custom course notes, and integrated into the lab exercises.

CPRE 488 is a senior-level course organized as three lecture hours and a three-hour lab session weekly. This was created in fall 2005 to bridge the gap between CPRE 211 and CPRE 588



Fig. 1. PowerBox: PowerPC-based platform.

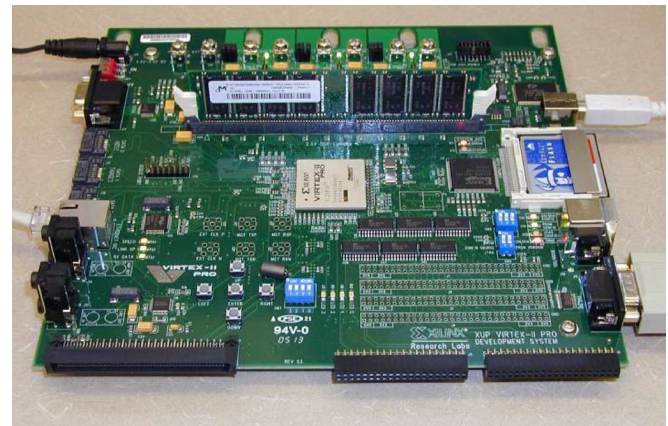


Fig. 2. Xilinx Virtex II Pro-based platform.

(summarized later). The goal was to introduce system-level design experiences for students and to incorporate new technology advancements. Prior to taking CPRE 488, students are expected to have studied computer organization and operating systems. The course investigates hardware and software design issues from a system-level perspective and also weaves in related advanced topics, such as software engineering methods and real-time scheduling policies. As in CPRE 211, the labs are a significant component and are integrated with classroom teaching. The lab platform hardware uses the Xilinx Virtex II Pro board from Digilent, Pullman, WA, shown in Fig. 2. The board includes a Xilinx field-programmable gate array (FPGA) chip, which has two PowerPC 405 processor cores, and extensive I/O capabilities. HW/SW development is supported by the Xilinx Platform Studio integrated development environment, including the embedded development kit (EDK) and the integral of square error (ISE) logic design toolset. The VxWorks real-time operating system (RTOS) is also supported. This platform and environment approximate the industry standard and expose students to the rich features and full complexity of contemporary embedded systems design. Lab experiences are oriented toward problem-based learning using two applications, digital cameras and MP3 players. Through the labs, students gain hands-on experience with performance analysis, profiling methods, hardware accelerator design, real-time scheduling, and system testing. CPRE 488 is described in detail in Section III.

TABLE I  
LEARNING MODELS AND 3C5I

| 5I Learning Steps | Bloom's Levels            | Problem-Based Learning               |
|-------------------|---------------------------|--------------------------------------|
| Introduction      | –                         | Problem, Conceptualization           |
| Illustration      | –                         |                                      |
| Instruction       | Knowledge   Comprehension | Process Skills, Subject Knowledge    |
| Investigation     | Application   Analysis    | Active Learning, Problem-Solving     |
| Implementation    | Synthesis   Evaluation    | Creative Thinking, Critical Thinking |

CPRE 588 is a three-credit graduate-level course that has been taught for a number of years, and is based on advances in the field and new research directions. The course focuses primarily on system-on-a-chip design methodologies and modeling languages for embedded computer systems, as well as various models of computation. Graduate students and qualified undergraduates take the course, which is also offered as a distance education course; the background of the students is thus much more diverse than in either CPRE 211 or CPRE 488. Since 2002, two system-level design languages, SpecC [15], [16] and SystemC [17], [18], have been used in CPRE 588 [19], [20]. SpecC includes a handful of explicit language constructs that directly address characteristics of embedded applications, which gives students a quick start into understanding the concepts of system-level modeling. A clear refinement methodology across multiple models and levels of abstraction is defined for SpecC. Many commercial tools used in industry support SystemC. Thus, a combination of SpecC and SystemC in CPRE 588 provides students with a balance of theory and practice. There are no scheduled laboratory sessions in CPRE 588. Students apply the SpecC methodology throughout the course, starting with the specification model of an embedded system and making a series of refinements to simulate and optimize an implementation of the system.

### B. Learning Model

This curriculum exemplifies one of the key challenges in computer engineering education: developing the educational context for new technologies. As new system design concepts and technologies emerge with the rapid advancement of the field, it is essential to develop and test new educational materials that teach and measure deep understanding, as recommended in *How People Learn* [22]. The achievement of depth of understanding through increasing levels of cognition was introduced in Bloom's taxonomy [23], [24]. Bloom's taxonomy includes a categorization of learning into six cognitive levels:

- knowledge – recalling previously learned information;
- comprehension – understanding the meaning;
- application – solving problems using information;
- analysis – examining information and making inferences;
- synthesis – creatively applying or integrating prior knowledge; and
- evaluation – making judgements about information and ideas.

The levels are hierarchal and increasingly complex, and each level is achieved before moving to a higher level. Knowledge and comprehension are typically viewed as lower-order thinking, while application, analysis, synthesis, and evaluation, as higher-order thinking. Other progressions have been defined for reaching deeper understanding. In problem-based learning (PBL), the problem drives the learning [25]. Before students learn some knowledge, they are given a problem. The problem is posed so that students discover that they need to learn some new knowledge before they can solve the problem. For example, in guided design, a case is posed, and small groups work cooperatively using structured problem-solving to make decisions. PBL often includes many principles that improve learning, such as active learning, cooperation, prompt feedback, diverse learning styles, and student empowerment and accountability.

The course developers have used a model called 3C5I that incorporates both Bloom's levels and elements of PBL [13]. The 3C5I model creates an educational context based on Concepts within Courses within a Curriculum (3C), and in each, progressing along the five learning steps of Introduction, Illustration, Instruction, Investigation, and Implementation (5I). Table I lists the 5I learning steps and their relationship to the Bloom and PBL learning models.

Both 3C5I and PBL include stages preceding Bloom's lowest order of knowledge. PBL may extend through to either Investigation or Implementation, as described in 3C5I, to differing degrees depending on the scope of the problem. In developing the embedded systems courses, targets were set for the level of learning to be accomplished in lecture versus laboratory and across labs. 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. The 3C5I model served as a structured approach to organizing classroom and laboratory learning for the series of courses in the curriculum. Using a learning model as the foundation facilitated the collaboration of the instructional team. Each of the courses has been developed, taught and/or enhanced by the authors over the past five years, resulting in a team of faculty and students engaged in the curriculum. This team effort led to higher-quality courses.

### III. COURSE DESIGN

#### A. Design Strategy

CPRE 488 was developed by the instructional team implementing the curriculum highlighted in Section II. An educational grant from the Rockwell Collins Foundation provided seed funding for the instructional laboratory. Contemporary practices informed course design; in particular, the “backwards design process” presented by Wiggins and McTighe gives three steps, starting with learning outcomes [28]:

- identify desired outcomes;
- determine acceptable evidence that outcomes have been achieved;
- plan learning experiences, instruction, and appropriate assessments.

The learning outcomes defined for CPRE 488 include:

- understanding the operational principles and technological advancements of embedded computer systems and their components;
- developing an ability to integrate embedded software, hardware, and operating systems to meet functional and performance requirements of embedded applications;
- developing an ability to use modern design methodologies and tools for developing and testing complex HW/SW systems;
- being familiar with system-level design concepts;
- understanding and applying techniques and tools for performance analysis.

These outcomes led to the following list of course topics: embedded microprocessors, embedded memory and I/O devices, component interfaces, embedded software, program development, basic compiler techniques, platform-based FPGA technology, hardware synthesis, design methodology, RTOS concepts, performance analysis, and optimizations.

With these outcomes and topics as a target, the instructional team used the 3C5I learning model to map out the course. The team met weekly over several months to brainstorm and plan the learning steps and experiences to support the outcomes and topics. A collaborative approach was used with work-in-progress recorded on large self-stick wall sheets. These wall sheets remained posted for reference during team-based development of the course materials. Each sheet was labeled with a week of the semester, topics were listed for the lecture and/or laboratory, and the expected level of learning (Introduction, Illustration, Instruction, Investigation, Implementation) was identified in each case. Thus, much as described in Section II, the team created a plan that progressed through learning steps for each of the topics and related concepts over one or more lessons and throughout the course. For example, the team decided that the learning goal for an RTOS concept would be at the level of investigation. Team members then determined when and how to provide each of introduction, illustration, instruction, and investigation in lectures and labs. In addition, two systems, a digital camera and an MP3 player, were selected early on to provide intensive experiences through application-oriented labs. Learning opportunities associated with these labs were carefully reviewed and integrated. Thus,

the course plan was motivated by goals for student learning (cognitive levels, relationships, problem-based inquiry, etc.) and not simply coverage of content.

The instructional team and the course development both benefited from the type of teaming described by Bess and others in [29], which defines a diversity of roles or tasks in the process of teaching. The traditional view of team teaching involves faculty having different disciplinary knowledge so as to teach a complex subject. In Bess’ view of team teaching, a team of specialists is formed based on process knowledge, i.e., the process of teaching. Members take on specialized roles, such as preparatory roles (e.g., research), classroom roles (e.g., lecturing), and facilitating roles (e.g., assessment). Thus, the teaching team is a self-managing team comprised of method specialists who assume shared responsibility for the teaching enterprise but who take leadership of and are individually accountable for particular functions [30]. In fact, the authors of this paper have constituted such a team for CPRE 488, with one member responsible for lectures, another for researching and testing new lab technologies and tools, another for evaluating the course, and so on.

#### B. Course Description

The resulting organization of the course is shown in Table II for a 15-week semester. There are two lecture sessions and one lab session per week. During the semester, there are nine labs and a lab project, all supervised by a trained teaching assistant. The number of homework sets has varied in different offerings of the course, and eight are shown in the table. Midterm and final exams were also scheduled.

The linkages among elements in the course are built into the plan. For example, performance is introduced in Lecture 5; illustration, instruction, and investigation related to performance follow in Lab 3 and Lecture 7; further investigation and implementation are done in Labs 4–6; and then Lecture 15 revisits the subject. Integration is achieved through flow of information from lecture to lab and back to lecture.

The nine regular lab sessions fall into three sets, as shown in Table III: Set 1—Introductory; Set 2—Digital Camera; and Set 3—MP3 Player. Sets 2 and 3 are motivated by the two applications. Each set emphasizes different topics, as shown in the table, and takes students through a series of learning steps.

Because laboratory learning will be considered in more detail in Section IV, the content of each lab is summarized later. All labs are available online [11].

- Lab 1: Students are introduced to the Xilinx ISE/EDK toolset and the boards. Students study the system and write a simple flashing LED program. Xilinx documentation is referenced to gain familiarity for later labs.
- Lab 2: Students are given hardware cores to add to the design. An interrupt controller is added and connected to the push-buttons available on the board. Students write a program that calls an interrupt handler when the button is pressed. This introduces the interrupt structure used in the hardware.
- Lab 3: Students migrate functional modules in the embedded system from software to hardware. This is called hardware acceleration. Students explore performance issues associated with computation and memory elements.

TABLE II  
COURSE PLAN FOR CPRE 488 (FALL 2006)

| Week | Labs  | Lectures   |
|------|---|--|
| 1    |   | 1. Introduction - Embedded Computing Systems<br>2. Design Process of Embedded Computing Systems  |
| 2    | 1. Introduction to Platform and Toolset                 | 3. I/O Interface and Interrupt Systems<br>4. (continued)<br><i>Homework 1</i>  |
| 3    | 2. Hardware/Software Interface                          | 5. CPU Performance and Power Consumption<br>6. (continued)   |
| 4    | 3. Hardware Acceleration                                | 7. Accelerated Systems<br>8. Object-Oriented Design and Unified Modeling Language<br><i>Homework 2</i>   |
| 5    | 4. Profiling (Digital Camera Part I)                    | 9. FPGA Introduction<br>10. Embedded Processor Technology<br><i>Homework 3</i>   |
| 6    | 5. Creating Custom IP Part I (Digital Camera Part II)   | 11. Common I/O Devices<br>12. Bus and Memory<br><i>Homework 4</i>  |
| 7    | 6. Creating Custom IP Part II (Digital Camera Part III) | 13. Basic Compiler Techniques<br>14. Program Design, Representation, and Assembly and Linking<br><i>Homework 5</i>                             |
| 8    | 7. Real-time Operating Systems (MP3 Player Part I)      | 15. Analyses and Optimizations of Performance, Energy, and Program Size  |
| 9    | 8. System Design using an RTOS (MP3 Player Part II)     | 16. Analyses and Optimizations of Performance, Energy, and Program Size (continued)<br>17. Program Validation and Testing<br><i>Homework 6</i> |
| 10   | 9. Networking in VxWorks                                | 18. Embedded Operating Systems<br>19. (continued)<br><i>Homework 7</i>   |
| 11   | Project: Application Selection                          | 20. Process Scheduling Policies<br>21. Real-time Scheduling Policies<br><i>Homework 8</i>  |
| 12   | Project: Requirements and Analysis                      | 22. IPC and Power Management<br>23. Embedded Operating Systems   |
| 13   | Project: Development                                    | 24. Networks for Embedded Systems<br>25. (continued)   |
| 14   | Project: Development                                    | 26. Design Methodologies<br>27. Project Presentations (by students)  |
| 15   | Project: Testing and Demonstration                      | 28. System Analysis and Architecture Design<br>29. Course Review   |

- Lab 4: Students are given a software implementation of a JPEG encoder, which reads an uncompressed image from a camera, encodes it, and saves it to a compact flash card. A set of requirements for the HW/SW system is also given, specifically a time constraint that requires custom hardware. Profiling methods are used to identify candidate functions to optimize.
- Lab 5: Students use the profile information collected in Lab 4 to improve the performance of the digital camera application. Students use software optimizations only.
- Lab 6: Students continue with optimizing the digital camera application, now adding custom hardware to meet the timing constraint.
- Lab 7: Students are introduced to VxWorks, a commercial RTOS. Students compile the VxWorks kernel and modify a program to illustrate concepts in RTOSs, such as tasks, inter-task communication, task scheduling, and interrupts.
- Lab 8: Students are provided with code for an MP3 player system developed for the VxWorks RTOS. The system

plays MP3 files, but the sound quality is very poor. Students must implement improvements to the player to obtain CD-quality sound.

- Lab 9: Students implement a Web server using basic networking features in VxWorks.

As previously described, these labs were developed with a progression of learning in mind. At the end of the first set of labs, students are well-acquainted with the hardware platform and development environment. In Lab 3, the last lab in the introductory set, the effect of memory on system performance is analyzed through matrix multiplication. Students implement matrix multiplication with and without cache support. A coprocessor is then used to demonstrate speedup using custom hardware. Hardware acceleration, introduced in Lab 3, is central to the next set of labs. In Lab 6, the last lab in the second set, students must apply the techniques from Lab 3 to meet system performance requirements in a complex application (digital camera). The last set of labs also revisits earlier material. In Lab 1, a simple application involving LEDs and push-buttons was used to let students

TABLE III  
CPRE 488 LABS

| Set                 | Lab |  | Software | Hardware | Performance Analysis | RTOS | Networking |
|---------------------|-----|--|----------|----------|----------------------|------|------------|
| 1:<br>Intro.        | 1   | Platform and Toolset                             | x        |          |                      |      |            |
|                     | 2   | Hardware/Software Interface                      | x        | x        |                      |      |            |
|                     | 3   | Hardware Acceleration                            | x        | x        | x                    |      |            |
| 2:<br>Camera        | 4   | Profiling (Digital Camera Part I)                | x        | x        | x                    |      |            |
|                     | 5   | Software Optimization (Digital Camera Part II)   | x        | x        | x                    |      |            |
|                     | 6   | Hardware Optimization (Digital Camera Part III)  | x        | x        | x                    |      |            |
| 3:<br>MP3<br>Player | 7   | Real-time Operating Systems (MP3 Player Part I)  | x        |          | x                    | x    |            |
|                     | 8   | System Design using an RTOS (MP3 Player Part II) | x        | x        | x                    | x    |            |
|                     | 9   | Networking in VxWorks                            | x        | x        | x                    | x    | x          |
| Project             |     |  | x        | x        | x                    | x    | x          |

practice with the new platform and environment. In Lab 7, a new RTOS layer is added to the environment, and the same simple application from Lab 1 is again used to let students see the new capabilities. Using the VxWorks RTOS in Lab 8, students configure an industry-standard audio codec to communicate with an MP3 decoder using interrupts. Lab 8 pulls together various skills from previous labs.

1) *CPRE 488 Content and Pedagogy*: A common problem for embedded system courses that use contemporary toolsets and conventional textbooks is synchronizing the labs and lectures. In CPRE 488, the Xilinx EDK/ISE development environment has two interdependent sets of tools, one for FPGA hardware design and another for software development. To configure and use such an environment, students must have breadth of knowledge and skills in both hardware and software. CPRE 488 adopted the textbook by Wolf [27], as it covers a diverse array of subject matter. However, the labs require knowledge not found in the textbook as well as an in-depth understanding of selected applications. Moreover, in the textbook, 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. Special attention was given to the organization of subjects in the lectures, clearly identifying relationships between lecture and lab, and progressively and iteratively developing student skills in the laboratory.

The instructional team reorganized the sequence of lectures so that the basics of some topics are introduced early in the semester. Within the first six weeks, students are exposed to embedded CPUs and memory components, common I/O devices, FPGA basics, hardware accelerators, and compiler techniques. This reorganization placed each topic within a realistic system context and helped prepare students for the lab exercises.

The system perspective in CPRE 488 is uniquely positioned in the integrated series between CPRE 211 and 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

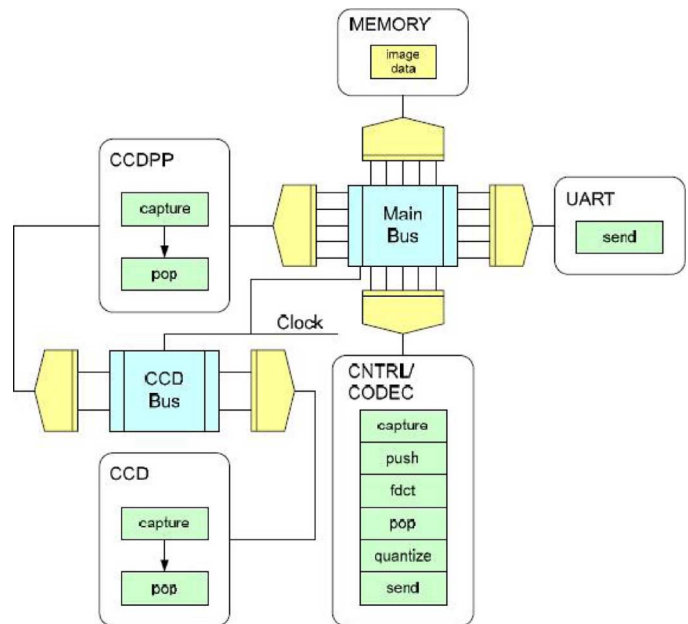


Fig. 3. SpecC communication model used in CPRE 588 (from [20]).

to an actual implementation. On the other hand, to bridge the gap with introductory courses such as CPRE 211, 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.

The digital camera application serves as an example to highlight the system perspective provided in CPRE 488, in contrast to the top-down approach of CPRE 588. The digital camera example described here is based on a simplified system model shown by Vahid and Givargis [26]. The camera system has only one function—to capture, process, and store images—yet it typifies common tasks done in embedded systems. In CPRE 588,

```

Flat profile:
Each sample counts as 0.0001 seconds.
%   cumulative   self   self   total
time seconds  seconds  calls  s/call  s/call  name
21.82    6.38    6.38    53     0.12    0.12  get_frame
10.74    9.52    3.14
 8.31   14.70    2.43
 5.07   17.90    1.48
 4.67   19.26    1.37
 3.60   22.57    1.05
 2.17   23.20    0.64
 2.02   24.41    0.59   9504    0.00    0.00  JpegEncoder

```

Fig. 4. Sample profiling output from CPRE 488 lab.

the digital camera example is used to investigate topics ranging from system modeling to HW/SW implementation. Students in CPRE 588 are first introduced to the specifications of the digital camera and to 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 the specification model to the implementation model, analyzing and performing refinements in this process. Fig. 3 depicts a SpecC communication model of the digital camera. In this model, tasks have been partitioned among processing elements and details about communication between tasks have been added. This system is represented at a more abstract level than in CPRE 488.

In CPRE 488, the digital camera lab addresses several learning objectives in the course, including developing an understanding of how to integrate hardware and software to meet the functional and performance requirements of embedded applications. Students are introduced to custom hardware components for the digital camera, specifically, the camera itself and its device interface. Students previously gained proficiency with the design platform and tools in the laboratory. Students follow a structured design process that creates a system having both software and hardware components. Students use profiling tools to investigate design alternatives that meet performance requirements for processing the image. In the CPRE 488 digital camera lab sequence, students conduct a comparative analysis of three design alternatives: software-only, software-only with optimizations, and mixed HW/SW. The analysis shows students 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. The design environment supports 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 Fig. 4, listing the number of calls to a function as well as the time spent in that function. These data are then used to guide the design refinements.

After the profiling exercise, students apply software techniques to improve the performance. This step is used to show that software techniques are available; however, the performance requirement in this exercise is carefully chosen so that software improvements are not sufficient. The students then identify which functions are candidates for implementation in hardware. This decision is made based on profiling data collected after implementing software optimizations for the image encoding. In the final step involving the digital camera, students integrate a custom hardware component into the system, profile the new system, and perform a comparative analysis of the three digital camera systems. The topical coverage and pedagogical approach highlighted in this example are the basis for a system-level design perspective in CPRE 488 that is not available in either CPRE 211 or CPRE 588.

2) *Comparison of CPRE 488 With Reference Curriculum:* Finally, CPRE 488 can be viewed in relation to the reference curriculum *Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering (CE2004)* [31]. Computer engineering educators may be familiar with this report, as it defines a body of knowledge (BOK) for computer engineering consisting of knowledge areas and units, each comprised of topics and learning outcomes. For example, the embedded systems knowledge area (CE-ESY) – one of 18 areas – covers 11 knowledge units. There is a unit on embedded microcontrollers (CE-ESY1). Topics are listed for this unit, such as structure of a basic computer system, polled I/O versus interrupt-driven I/O, etc. Corresponding learning outcomes are also listed for each unit (i.e., understand the CPU in the context of a complete system with I/O and memory, understand how memory system design affects program design and performance). The BOK provides a general framework for the content of CPRE 488.

Table IV presents the coverage of the CE-ESY knowledge area in CPRE 488, by identifying lectures, labs, and homework that include material related to a unit. All units except CE-ESY4 (low-power computing) are represented in every part of the course. There is varying coverage of both core and elective units, with emphasis on those that match the learning outcomes used as the basis for course design.

The instructional team has also critiqued the lab exercises with respect to specific *CE2004* learning outcomes, with attention paid to the following outcomes.

TABLE IV  
CPRE 488 COVERAGE OF THE CE2004 EMBEDDED SYSTEMS KNOWLEDGE AREA

| CE-ESY Knowledge Units                                   | Labs                | Lectures               | Homeworks     |
|--|---------------------|------------------------|---------------|
| CE-ESY0 History and Overview (core)                      | 1                   | 1                      | 1             |
| CE-ESY1 Embedded microcontrollers (core)                 | 1, 2, Project       | 4, 8, 9, 10            | 1, 2, 3, 4, 5 |
| CE-ESY2 Embedded programs (core)                         | 1, 2, 3, 4, Project | 11, 12, 13, 14         | 3, 6, 7       |
| CE-ESY3 Real time operating systems (core)               | 7, 8, 9             | 16, 17, 18, 19, 20, 21 | 8, 9          |
| CE-ESY4 Low power computing (core)                       |                     | 4, 13, 14, 20          | 2             |
| CE-ESY5 Reliable system design (core)                    | Project             | 15                     | 7             |
| CE-ESY6 Design methodologies (core)                      | All labs, Project   | 2, 6, 26, 27, 28       | 1, 3, 4       |
| CE-ESY7 Tool support (elective)                          | All labs            | 13, 14                 | 6             |
| CE-ESY8 Embedded multiprocessors (elective)              | 1, 2, 3, 4, 5, 6    | 3, 5, 7                | 2, 3          |
| CE-ESY9 Networked embedded systems (elective)            | 9                   | 22, 23, 24, 25         | 10            |
| CE-ESY10 Interfacing and mixed signal systems (elective) | 3                   | 9                      | 4             |

- CE-ESY1-2. Understand how the CPU talks to the outside world through devices.
- CE-ESY1-3. Understand how memory system design affects program design and performance.
- CE-ESY3-3. Understand major real-time scheduling policies.
- CE-ESY6-1. Understand why real-world projects are not the same as class projects.
- CE-ESY7-2. Understand how to use HW/SW tools to support system design methodology.
- CE-ESY8-2. Identify tradeoffs between CPUs and hard-wired logic in multiprocessors.
- CE-ESY8-3. Understand basic design techniques.
- CE-CAO8-1. Understand the factors that contribute to computer performance.
- CE-CSE8-2. Apply HW/SW codesign principles in situations of modest complexity.

This process has provided an additional means to benchmark the course and labs, as well as having prepared the team for future work on developing rubrics to provide students with feedback on their learning.

#### IV. ASSESSMENT

The design of the course emphasized laboratory learning and providing students with opportunities for higher-order thinking, based on the educational premise that student learning is improved as students progress through higher cognitive levels. Thus, a key question of interest to the instructional team was whether the lab activities provided learning opportunities at higher cognition levels. This section characterizes the learning experience in the laboratory using a technique to assess cognitive behavior. The results of this assessment reveal a striking pattern in the progression of cognitive skills expected in the three sets of labs.

#### A. Methodology

Following Ulmer and Torres [33], the FTCB [7], [36], a modified form of Bloom's taxonomy, was used to provide a framework to guide assessment of the level of cognitive behavior observed in the laboratory assignments completed by students in the class. Table V lists the seven levels of cognitive behavior in FTCB, in ascending order of cognitive sophistication, as: 1) knowledge of specifics, of ways and means of dealing with specifics, and of universals and abstracts; 2) translation; 3) interpretation; 4) application; 5) analysis; 6) synthesis (creativity); and 7) evaluation. In Bloom's taxonomy, 2) and 3) are treated as a single category (comprehension). The subsequent data analysis uses one-way analysis of variance (ANOVA), repeated measures ANOVA, and Pearson product-moment correlations [37], [38]. Results are reported in summary tables and figures, with interpretation.

#### B. Results

The FTCB was used to categorize cognitive behaviors observed in laboratory assignments over ten-point intervals. As a behavior was observed, a box was marked within the cognitive category (knowledge, translation, interpretation, application, analysis, synthesis, and evaluation). A behavior was recorded only once per ten-point interval, regardless of the number of times it occurred. For example, if a lab activity asks for an explanation three times in an interval, the box was marked only once. However, if a different explanation is asked for during three different intervals, a box was marked in all three intervals. Each level of cognitive behavior was recorded as it was seen by each of three observers. These results are summarized in Table VI.

Averaging across the ratings of the cognitive level of the nine labs by the three observers, 62.9% of all behaviors observed

TABLE V  
ELEMENTS OF FTCB USED TO ASSESS COGNITIVE BEHAVIOR  
DEMONSTRATED IN CLASS ASSIGNMENTS

|  |
|--|
| 1.1 Knowledge of Specifics                                   |
| 1. Reads   |
| 2. Spells  |
| 3. Identifies something by name                              |
| 4. Defines meaning of a term                                 |
| 5. Gives a specific fact                                     |
| 6. Tells about an event                                      |
| 1.2 Knowledge of ways and means of dealing with specifics    |
| 7. Recognizes symbol   |
| 8. Cites a rule  |
| 9. Gives chronological sequence                              |
| 10. Gives steps of process, describes method                 |
| 11. Cites trend  |
| 12. Names classification system or standard                  |
| 13. Names what fits given classification system or standard  |
| 1.3 Knowledge of universals and abstracts                    |
| 14. States generalized concept or idea                       |
| 15. States a principle, law, or theory                       |
| 16. Tells about organization or structure                    |
| 17. Recalls name of principle, law, or theory                |
| 2. Translation   |
| 18. Restates in own words or briefer terms                   |
| 19. Gives concrete examples of an abstract idea              |
| 20. Verbalizes from a graphic representation                 |
| 21. Translates verbalization into graphic form               |
| 22. Translates figurative statements into literal statements |
| 23. Translates foreign language into English or vice versa   |
| 3. Interpretation  |
| 24. Gives reason (tells why)                                 |
| 25. Shows similarities and differences                       |
| 26. Summarizes or concludes from observations of evidence    |
| 27. Shows cause and effect relationship                      |
| 28. Gives analogy, simile, metaphor                          |
| 29. Performs a directed task or process                      |
| 4. Application   |
| 30. Applies previous learning to new situations              |
| 31. Applies principle to new situation                       |
| 32. Applies abstract knowledge in a practical situation      |
| 33. Identifies, selects, and carries out a process           |
| 5. Analysis  |
| 34. Distinguishes fact from opinion                          |
| 35. Distinguishes fact from hypothesis                       |
| 36. Distinguishes conclusions from supporting statements     |
| 37. Points out unstated assumption                           |
| 38. Shows interaction or relation of elements                |
| 39. Points out particulars to justify conclusions            |
| 40. Checks hypotheses with given information                 |
| 41. Distinguishes relevant from irrelevant statements        |
| 42. Detects error in thinking                                |
| 43. Infers purpose, point of view, thoughts, feelings        |
| 44. Recognizes bias or propaganda                            |
| 6. Synthesis (Creativity)                                    |
| 45. Reorganizes ideas, materials, processes                  |
| 46. Produces unique communication, divergent idea            |
| 47. Produces a plan, proposed set of operations              |
| 48. Designs an apparatus                                     |
| 49. Designs a structure                                      |
| 50. Devises a scheme for classifying information             |
| 51. Formulates hypotheses, intelligent guesses               |
| 52. Makes deductions from abstract symbols, properties       |
| 53. Draws inductive generalization from specifics            |
| 7. Evaluation  |
| 54. Evaluates something from evidence                        |
| 55. Evaluates something from criteria                        |

were higher-order behaviors. Of the higher-order cognitive behaviors, the most common was analysis (23.7%) and the

least common was evaluation (4.5%). An average of 37.1% of all behaviors observed were lower-order cognitive behaviors. Of the lower-order cognitive behaviors, the most common was interpretation (19.6%) and the least common was knowledge (8.3%). Of all seven categories, the most commonly observed cognitive behavior was analysis, and the least common was evaluation.

A Pearson chi-square test ( $\chi^2 = 36.957$ ,  $df = 12$ ,  $p < 0.001$ ) demonstrated that there is a statistically significant difference in the ratings provided by the three observers. This interrater difference is clear from Table VI, which shows a major difference between Observer 2 (57.9% lower-order) and Observers 1 and 3 (28.7% and 32.7% lower-order, respectively). Such observer differences are also evident in the literature [32]–[35].

Table VII and Fig. 5 compare the mean cognitive ratings between labs, with differences shown numerically and visually, respectively. The cognitive levels range from 1 for knowledge to 7 for evaluation. A one-way ANOVA demonstrates that the differences between labs are statistically significant ( $F = 2.730$ ,  $p = 0.006$ ); this result is confirmed by robust tests (Welch and Brown-Forsythe). Multiple comparisons (Bonferroni) appropriate to the circumstances of equal variances across labs (Levene approximate  $F = 1.625$ ,  $df = 8$ ,  $p = .116$ ) shows a significantly higher mean for Lab 6 compared to Lab 4; no other pairwise differences are significant. A Pearson product-moment correlation of time (coded 1 for Lab 1, to 9 for Lab 9) results in a positive ( $r = .443$ ), but not statistically significant ( $p = .233$ ) relationship; the general trend is toward increased sophistication, measured by higher mean cognitive level, over time.

The differences between labs are consistent with the design of the labs. Labs 1–3 form an introductory set, with more advanced skills introduced in each lab. Labs 4–6 are based on the digital camera application, with increasing complexity. Because these labs are based on a real application, one might expect this set to reach a higher level of cognition than the first set of three labs. Labs 7–9 form a third set, with lab 7 being somewhat independent, introducing students to an advanced topic and tools. One might expect relatively higher-order thinking skills on lab 7 compared to labs 1 and 4. Labs 8 and 9 then build from lab 7 on another real application (MP3 player). One might have expected lab 8 to be rated higher, and its rating deserves a closer look. However, the results accurately portray the fact that students need to make a bigger jump in the level of tasks being done from start to finish on labs 4–6, compared to labs 7–9. The pattern in the progression of cognitive skills is notable. Students progress from lower-order thinking skills to higher-order skills in each set of labs, and start each set at a relatively low skill level.

While a detailed study of student performance is beyond the scope of this paper, Fig. 6 compares the mean percentage student scores between labs. It is interesting to compare the patterns in the figures. There is a relationship between student performance in the lab and the cognitive level of the lab. The correlation between the mean percentage student score on each lab, and the lab's mean cognitive level of  $r = -.763$ , is significant ( $p = 0.017$ ). This result is a reasonable outcome; as the semester unfolds, course material tends to become more

TABLE VI  
PERCENTAGE BY COGNITIVE LEVEL ON NINE LABS, FOR EACH OF THREE OBSERVERS

| Cognitive Level   | Average (%) | Observer 1 (%) | Observer 2 (%) | Observer 3 (%) |
|-------------------|-------------|----------------|----------------|----------------|
| Lower-Order       | 37.1        | 28.7           | 57.9           | 32.7           |
| 1. Knowledge      | 8.3         | 8.3            | 11.0           | 7.1            |
| 2. Translation    | 9.2         | 2.8            | 13.7           | 11.5           |
| 3. Interpretation | 19.6        | 17.6           | 34.2           | 14.1           |
| Higher-Order      | 62.9        | 71.3           | 42.1           | 67.3           |
| 4. Application    | 13.9        | 9.3            | 12.3           | 17.9           |
| 5. Analysis       | 23.7        | 27.8           | 17.8           | 23.7           |
| 6. Synthesis      | 20.8        | 30.6           | 9.6            | 19.2           |
| 7. Evaluation     | 4.5         | 3.7            | 1.4            | 6.4            |

TABLE VII  
MEAN COGNITIVE LEVEL, BY LAB

| Lab   | Number of behaviors observed | Mean Level | Standard Deviation | Standard Error | 95% Confidence Interval for Mean |      | Minimum Level | Maximum Level |
|-------|------------------------------|------------|--------------------|----------------|----------------------------------|------|---------------|---------------|
|       |                              |            |                    |                |                                  |      |               |               |
| 1     | 31                           | 3.58       | 1.822              | .327           | 2.91                             | 4.25 | 1             | 6             |
| 2     | 39                           | 4.15       | 1.740              | .279           | 3.59                             | 4.72 | 1             | 7             |
| 3     | 51                           | 4.57       | 1.500              | .210           | 4.15                             | 4.99 | 1             | 7             |
| 4     | 39                           | 3.44       | 1.847              | .296           | 2.84                             | 4.03 | 1             | 7             |
| 5     | 38                           | 4.37       | 1.584              | .257           | 3.85                             | 4.89 | 1             | 7             |
| 6     | 19                           | 4.95       | 1.353              | .310           | 4.30                             | 5.60 | 2             | 7             |
| 7     | 66                           | 4.08       | 1.439              | .177           | 3.72                             | 4.43 | 1             | 6             |
| 8     | 36                           | 4.11       | 1.801              | .300           | 3.50                             | 4.72 | 1             | 7             |
| 9     | 18                           | 4.67       | 1.534              | .362           | 3.90                             | 5.43 | 2             | 6             |
| Total | 337                          | 4.16       | 1.662              | .091           | 3.98                             | 4.34 | 1             | 7             |

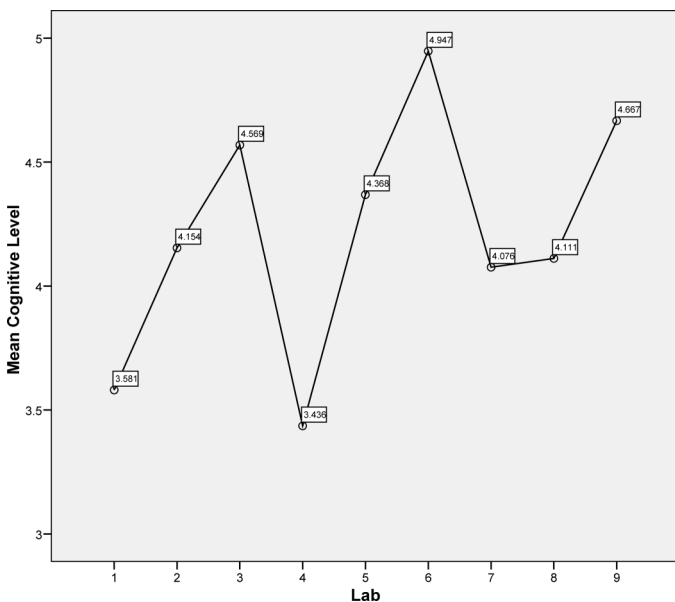


Fig. 5. Plot of mean cognitive level, by lab (from Table VII).

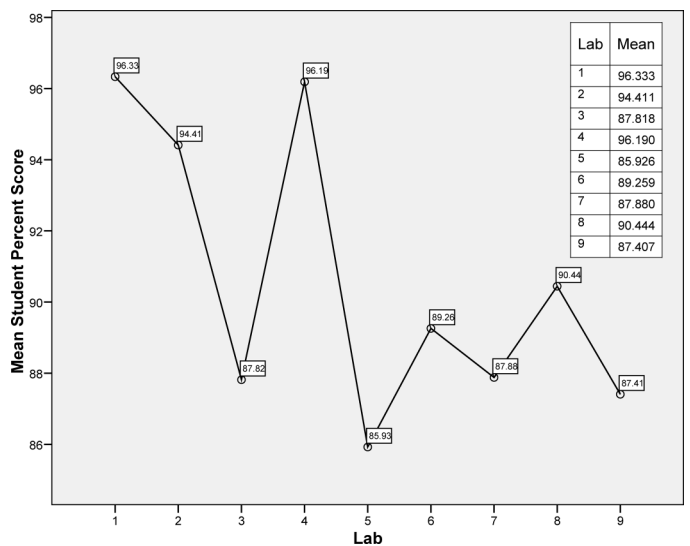


Fig. 6. Plot of mean student score (percentage), by lab.

detailed and more difficult, as verified by the previous finding that the mean cognitive level of the labs generally increases

over the course of the semester, so students confront conceptually more challenging material. This pattern is confirmed by a repeated measures ANOVA, which shows significant changes over time in the mean percentage of student scores on the

TABLE VIII  
TOTAL COGNITIVE WEIGHTED SCORES FOR EACH LAB AND ALL NINE LABS COMBINED

| Lab                   | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | All   |
|-----------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Knowledge             | 22.6  | 12.8  | 3.9   | 20.5  | 5.3   | 0.0   | 1.5   | 8.3   | 0.0   | 8.3   |
| Translation           | 6.5   | 10.3  | 3.9   | 10.3  | 5.3   | 5.3   | 12.1  | 13.9  | 16.7  | 9.2   |
| Interpretation        | 16.1  | 7.7   | 17.6  | 30.8  | 21.1  | 10.5  | 28.8  | 19.4  | 5.6   | 19.6  |
| Application           | 19.4  | 10.3  | 13.7  | 2.6   | 18.4  | 15.8  | 16.7  | 13.9  | 16.7  | 13.9  |
| Analysis              | 16.1  | 38.5  | 39.2  | 20.5  | 23.7  | 31.6  | 16.7  | 8.3   | 16.7  | 23.7  |
| Synthesis             | 19.4  | 17.9  | 9.8   | 10.3  | 18.4  | 26.3  | 24.2  | 33.3  | 44.4  | 20.8  |
| Evaluation            | 0.0   | 2.6   | 11.8  | 5.1   | 7.9   | 10.5  | 0.0   | 2.8   | 0.0   | 4.5   |
| Total Cognitive Score | 29.55 | 34.01 | 36.16 | 28.49 | 35.02 | 39.47 | 33.56 | 34.00 | 38.63 | 33.87 |

nine labs ( $Wilks' \lambda = .632$ ,  $F = 2.695$ ,  $p = 0.019$ ,  $partial \eta squared = .368$ ).

The differences in student scores are interesting relative to the sets of labs. The slightly higher score on lab 8 might reflect that students were not challenged as much as expected, especially noting the comparable cognitive level compared to lab 7. Further evaluation of student performance in this context is a promising area for future work.

Finally, a commonly-used measure with FTCB data is the total cognitive weighted score [35]. This is calculated as the weighted average of the percentage distribution of observed behaviors, where the weights are .10 for knowledge, .20 for translation, .25 for interpretation, .30 for application, .40 for analysis, and .50 for both synthesis and evaluation. The results for this calculation are shown in Table VIII. The lab assignments were found to have a total cognitive weighted score of 33.87, indicating an average cognitive level for laboratory activities above the application level. As with the previous results for mean cognitive score, there is a statistically significant negative correlation between the total cognitive weighted scores and mean student performance on the labs ( $r = -.755$ ,  $p = 0.019$ ). This finding confirms the conclusion that assignments requiring more sophisticated thinking by students enhance the ability of the labs as instruments measuring student outcomes to discriminate between students who are "getting it" at a higher cognitive level and those who are not.

## V. CONCLUSION

This paper began by citing a concern: "The need for a better understanding of the teaching/learning process in the laboratory is evident." Although far from complete, this paper has shed light on connected learning and higher-order learning through well-designed courses and labs in an embedded systems curriculum. This paper has explored the problem raised by Graham by considering not only the "what" and "how" of laboratory instruction, but also the "why," that is, understanding the learning experience in the laboratory. The assessment of cognitive levels using FTCB has not only provided insights into CPRE 488 and the cognitive depth of teaching and learning, but also led to greater reflection by the instructional team. For example, what patterns of lower-order and higher-order learning should

be achieved in a lab? In a course? In a curriculum? Ulmer makes an interesting observation on the difference between lower-order thinking and higher-order thinking [32]:

"... the difference is influenced by prior knowledge held by the learner. What may require higher-order thinking by one learner may require lower-order thinking by another learner. Arguably, what may require higher-order thinking by a learner today may not require the same level of thinking tomorrow. ... Lewis and Smith stated that the varying levels of the learner requires teaching be interwoven at different levels of cognition (cited as [39] by [32])."

This observation seems particularly relevant to a dynamic field such as computer engineering, and reinforces the need for learner-centered approaches informed by cognitive models.

### A. Future Work

These are the five general directions to pursue following the work reported in this paper.

- 1) Evaluate the assessment results to identify ways to improve the courses and enhance student learning.
- 2) Use the results to guide development of rubrics that provide students with feedback on their learning.
- 3) Perform more assessment and analysis to gain a broader and/or deeper understanding of the issues.
- 4) Share results, including the FTCB instrument, beyond the instructional team so as to impact other areas of electrical and computer engineering curricula in the department.
- 5) Conduct formal educational research in the learning sciences on laboratory learning.

Several specific activities that would have immediate value to teaching and learning in the course include: analysis of classroom learning and comparison with lab learning; measurement of student performance on learning outcomes in relation to statistical results of this study; and selected refinements to lab exercises to engage students in higher-order thinking.

### B. Related Work

A special issue of the *ACM Transactions on Embedded Computing Systems* in 2005 highlighted a number of educational

programs related to this work. At the University of California, Berkeley, the embedded system design education program consists of undergraduate and graduate coursework [43]. New advanced graduate courses are under development to support research results. One sophomore course in the area exists. Efforts are currently underway to develop a junior/senior-level course in mixed HW/SW systems design, similar to CPRE 488. There are also plans to develop courses based on the theoretical foundations of embedded systems design.

Carnegie Mellon University, Pittsburgh, PA, differentiates between the many application areas of embedded computer systems, and aims to give a broad exposure to these areas within its undergraduate curriculum [44]. 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. An embedded systems curriculum at the University of Waterloo, Waterloo, ON, Canada, identifies 16 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 [45]. Embedded systems education at Vanderbilt University, Nashville, TN, takes a model-based approach [46]. Technical elective courses were added to introduce a specialization in embedded systems design.

The European Artist Education Group identifies several challenges for developing a curriculum in embedded systems, as well as key bodies of knowledge that should be included [47]. The group emphasizes the need for lab experiences to strengthen understanding of core themes, which has been one of the drivers of course development at Iowa State. The bodies of knowledge are targeted to an entire graduate curriculum and are broad in scope. In contrast, this paper reports on a stream of embedded courses spanning undergraduate to graduate levels and, thus, is bounded by what can be covered in a few semesters' time. Lastly, a didactic analysis of embedded systems education suggests that the subject is effectively taught through functional examples in an interactive setting [48]. The lab sequence for CPRE 488 provides two such in-depth examples that allow students to gain a deeper understanding of the material.

#### ACKNOWLEDGMENT

The authors would like to thank J. Schneider, A. Larson, J. Boyd, and R. Walstrom for their contributions to the development of the laboratory.

#### REFERENCES

- [1] P. C. Wankat and F. S. Oreovicz, *Teaching Engineering*. New York, NY: McGraw-Hill, 1993.
- [2] L. D. Feisel and G. D. Peterson, "A colloquy on learning objectives for engineering education laboratories," in *Proc. Amer. Soc. for Engineering Education Annu. Conf.*, Montreal, QC, Canada, 2002, CD-ROM.
- [3] L. D. Feisel, G. D. Peterson, O. Armas, L. Carter, A. Rosa, and W. Worek, "Learning objectives for engineering education laboratories," in *Proc. Frontiers in Education Conf.*, Boston, MA, 2002, p. F1D-1.
- [4] W. C. Newstetter and J. Turns, "Looking for convergence: Laboratory learning and classroom learning," in *Proc. Frontiers in Education Conf.*, Boulder, CO, 2003, p. T4H-1.
- [5] G. D. Peterson and L. D. Feisel, "e-Learning: The challenge for engineering education," in *Proc. ECI Conf. e-Technologies in Engineering Education*, Davos, Switzerland, 2002 [Online]. Available: <http://services.bepress.com/eci/etechnologies>
- [6] R. Graham, "Needed: A Theory of Laboratory Instruction," in *The Undergraduate Engineering Laboratory*. New York: Engineering Foundation, 1983.
- [7] J. N. Webb, *The Florida Taxonomy of Cognitive Behavior: A Working Manual*. Tuscaloosa, AL: Univ. of Alabama, 1968.
- [8] M. Bezdek, D. Helvick, R. Mercado, D. Rover, A. Tyagi, and Z. Zhao, "Developing and teaching an integrated series of courses in embedded computer systems," in *Proc. Frontiers in Education Conf.*, San Diego, CA, Oct. 2006, pp. T1E19–24.
- [9] D. Helvick, R. Mercado, Z. Zhang, and D. T. Rover, "Reflections on implementing and teaching an advanced undergraduate course in embedded systems," in *Proc. Int. Conf. Microelectronic Systems Education*, San Diego, CA, Jun. 2007, pp. 5–6.
- [10] CPRE 211, Microcontrollers and Digital Systems Design [Online]. Available: <http://class.ece.iastate.edu/cpre211>
- [11] CPRE 488, Embedded Systems Design [Online]. Available: <http://class.ece.iastate.edu/cpre488>
- [12] CPRE 588, Embedded Computer Systems [Online]. Available: <http://class.ece.iastate.edu/cpre588>
- [13] A. Striegel and D. Rover, "Problem-based learning in an introductory computer engineering course," in *Proc. Frontiers in Education Conf.*, Boston, MA, 2002, pp. F1G7–12.
- [14] A. Striegel and D. Rover, "Enhancing student learning in an introductory embedded systems laboratory," in *Proc. Frontiers in Education Conf.*, Boston, MA, 2002, pp. T1D7–12.
- [15] SpecC Technology Open Consortium [Online]. Available: <http://www.specc.org>
- [16] D. Gajski, J. Zhu, R. Domer, A. Gerstlauer, and S. Zhao, *SpecC: Specification Language and Methodology*. New York: Springer, 2000.
- [17] SystemC [Online]. Available: <http://www.systemc.org>
- [18] *Design Automation Standards Committee of the IEEE Computer Society, IEEE Standard SystemC Language Reference Manual*, IEEE Standard 1666-2005, 2006.
- [19] J. Schneider, M. Bezdek, Z. Zhang, and D. T. Rover, "A platform FPGA-based hardware-software undergraduate laboratory," in *Proc. Int. Conf. Microelectronic Systems Education*, Anaheim, CA, 2005, pp. 53–54.
- [20] R. Walstrom, "System level design refinement using SystemC," M.S. thesis, Elect. Comput. Eng. Dept., Iowa State Univ., Ames, 2004.
- [21] K. Sakiyama, P. Schaumont, D. Hwang, and I. Verbauwhede, "Teaching trade-offs in system-level design methodologies," in *Proc. Int. Conf. Microelectronic Systems Education*, Anaheim, CA, 2003, pp. 62–63.
- [22] J. D. Bransford, A. L. Brown, and R. R. Cocking, *How People Learn: Brain, Mind, Experience, and School*. Washington, DC: National Academies Press, 2000.
- [23] B. S. Bloom and D. Krathwohl, *Taxonomy of Educational Objectives: The Classification of Educational Goals, Handbook I: Cognitive Domain*. White Plains, NY: Longman, 1956.
- [24] L. Anderson and D. Krathwohl, *A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, 2nd ed. Boston, MA: Allyn & Bacon, 2001.
- [25] D. Woods, "Problem-based learning: How to gain the most from PBL," Hamilton, ON, 1994 [Online]. Available: <http://chemeng.mcmaster.ca/pbl/pbl.htm>
- [26] F. Vahid and T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*. Hoboken, NJ: Wiley, 2002.
- [27] W. Wolf, *Computers as Components: Principles of Embedded Computing System Design*. San Francisco, CA: Morgan Kaufmann, 2001.
- [28] G. Wiggins and J. McTighe, *Understanding by Design*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1998.
- [29] J. L. Bess, *Teaching Alone/Teaching Together: Transforming the Structure of Teams for Teaching*. San Francisco, CA: Jossey-Bass, 2000.
- [30] D. T. Rover, "Taking our own advice: Team teaching," *J. Eng. Educ.*, vol. 91, no. 3, pp. 265–266, Jul. 2002.
- [31] "The joint task force on computing curricula," Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering, IEEE Computer Society, Association for Computing Machinery, Jul. 2006.
- [32] J. D. Ulmer, "An assessment of the cognitive behavior exhibited by secondary agriculture teachers," Ph.D. dissertation, Agricultural Education Dept., Univ. Missouri, Columbia, 2005.

- [33] J. D. Ulmer and R. M. Torres, "A classroom assessment of agriculture teachers' cognitive behaviors," in *2007 Proc. AAAE Research Conf.*, Minneapolis, MN, vol. 34, pp. 123–137.
- [34] A. B. Smith, G. Ward, and J. S. Rosenshein, "Improving instruction by measuring teacher discussion skills," *Amer. J. Phys.*, vol. 45, no. 1, pp. 83–87, Jan. 1977.
- [35] M. S. Whittington and L. H. Newcomb, "Aspired cognitive level of instruction, assessed cognitive level of instruction and attitude toward teaching at higher cognitive levels," *J. Agricult. Educ.*, vol. 34, no. 2, pp. 55–62, 1993.
- [36] B. B. Brown, R. Ober, R. Soar, and J. N. Webb, "Florida taxonomy of cognitive behavior: Directions," Univ. Florida, Gainesville, 1967, unpublished.
- [37] A. Agresti and B. Finlay, *Statistical Methods for the Social Sciences*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 1977.
- [38] B. Tabachnick and L. Fidell, *Using Multivariate Statistics*, 5th ed. Boston, MA: Allyn & Bacon, 2007.
- [39] A. Lewis and D. Smith, "Defining higher-order thinking," *Theory Into Practice*, vol. 32, pp. 131–137, 1993.
- [40] A. McGettrick, M. D. Theys, D. L. Soldan, and P. K. Srimani, "Computer engineering curriculum in the new millennium," *IEEE Trans. Educ.*, vol. 46, no. 4, pp. 456–462, Nov. 2003.
- [41] R. L. Traylor, D. Heer, and T. S. Fiez, "Using an integrated platform for learning to reinvent engineering education," *IEEE Trans. Educ.*, vol. 46, no. 4, pp. 409–419, Nov. 2003.
- [42] T. Roppel, "An interdisciplinary laboratory sequence in electrical and computer engineering: Curriculum design and assessment results," *IEEE Trans. Educ.*, vol. 43, no. 2, pp. 143–152, May 2000.
- [43] A. L. Sangiovanni-Vincentelli and A. Pinto, "An overview of embedded system design education at Berkeley," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 472–499, Aug. 2005.
- [44] P. Koopman *et al.*, "Undergraduate embedded system education at Carnegie Mellon," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 500–528, Aug. 2005.
- [45] R. E. Seviara, "A curriculum for embedded system engineering," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 569–586, Aug. 2005.
- [46] J. Sztipanovits *et al.*, "Introducing embedded software and systems education and advanced learning technology in an engineering curriculum," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 549–568, Aug. 2005.
- [47] P. Caspi *et al.*, "Guidelines for a graduate curriculum on embedded software and systems," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 587–611, Aug. 2005.
- [48] M. Grimheden and M. Törngren, "What is embedded systems and how should it be taught? – Results from a didactic analysis," *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 3, pp. 633–651, Aug. 2005.

**Diane T. Rover** (S'85–M'90–SM'01) received the B.S. degree in computer science, and the M.S. and Ph.D. degrees in computer engineering from Iowa State University, Ames, in 1984, 1986, and 1989, respectively.

She is currently Associate Dean for Academic and Student Affairs in the College of Engineering, Iowa State University. Since 2001, she has been a Professor in the Department of Electrical and Computer Engineering at Iowa State. From 1991–2001, she was Assistant Professor and Associate Professor in the Department of Electrical and Computer Engineering, Michigan State University, East Lansing. From 1997 to 2000, she served as Director of the undergraduate program in computer engineering. She also served as Interim Department Chair in the Department of Electrical and Computer Engineering from 2000 to 2001. At Iowa State, she served as Associate Chair for undergraduate education in the Department of Electrical and Computer Engineering from 2003 to 2004. She was a Research Staff Member in the Scalable Computing Laboratory at the Ames Laboratory under a U.S.-D.O.E. Postdoctoral Fellowship from 1989 to 1991. Her teaching and research has focused on the areas of digital logic design, hardware/software systems, reconfigurable hardware, integrated program development and performance environments for parallel and distributed systems, visualization, performance monitoring and evaluation, and engineering education.

Dr. Rover is a member of the IEEE Computer Society, the IEEE Education Society, and the American Society for Engineering Education (ASEE). Since 2006, she has been a member of the IEEE Committee on Engineering Accreditation Activities (CEAA). Since 2002, she has been an IEEE ABET/EAC Program Evaluator in computer engineering. She has served as senior associate editor for the academic bookshelf for the *ASEE Journal of Engineering Education* since 2000.

**Ramon A. Mercado** (S'04) received the B.S. degree in electrical engineering from the University of Puerto Rico, Rio Piedras, in 2004. He is working towards the Ph.D. degree in computer engineering at Iowa State University (ISU), Ames.

He is currently a U.S. Department of Education GAANN Fellow and Research Assistant in the Department of Electrical and Computer Engineering at ISU. He was a Teaching Assistant for CPRE 488 and CPRE 588 in the Department of Electrical and Computer Engineering in 2006. In summer 2004, he was a Research Assistant at the University of Michigan, Ann Arbor, where he worked on application code development for an environmental monitoring device. He has held several internships at Texas Instruments, including as Characterization Engineer in 2001 and 2003, and as System Level Test Engineer in 2005. His research interests involve system-level design methodologies for embedded systems, communication-centric design space exploration, and engineering education.

Mr. Mercado received the Lockheed Martin Fellowship in 2004. He has also received an ISU Teaching Excellence Award.

**Zhao Zhang** (S'97–M'04) received the B.S. and M.S. degrees from Huazhong University of Science and Technology, Wuhan, Hubei, China, and the Ph.D. degree from the College of William and Mary, Williamsburg, VA, in 1991, 1994, and 2002, respectively, all in computer science.

He is an Assistant Professor in the Department of Electrical and Computer Engineering at Iowa State University, Ames. His research interests include computer architecture and parallel and distributed systems.

Dr. Zhang is a member of the Association for Computing Machinery and the IEEE Computer Society.

**Mack C. Shelley** received the B.A. degree in economics and international studies from American University, Washington, DC, and the M.S. degree in economics and Ph.D. degree in political science from the University of Wisconsin, Madison, in 1972, 1973, and 1977, respectively.

He is a University Professor in the Departments of Statistics and Political Science (Public Policy and Administration Program), Iowa State University, Ames. From 2003 to 2007, he served as the Director of the Research Institute for Studies in Education at Iowa State. He began his academic career as an Assistant Professor in the Department of Political Science at Mississippi State University, Starkville, in 1977. He serves regularly as a statistical consultant and has worked with funding from numerous federal agencies, state agencies, and nonprofit organization, including the National Science Foundation and the U.S. Department of Education. He is the author of 10 books, 19 book chapters, 82 journal articles, and over 200 other publications. His research, external funding, and teaching focus is on statistical methods and their applications to public policy and program evaluation.

Dr. Shelley is a member of the American Statistical Association, American Educational Research Association, National Council on Measurement in Education, and Association for the Study of Higher Education. He has served as coeditor of the *Policy Studies Journal* (1993–2002), a member of the Editorial Advisory Board for *TESOL Quarterly* (2003–05), and associate editor of the *Journal of Information Technology & Politics* (2006–present). He has won awards for research, teaching, and professional practice.

**Daniel S. Helvick** (S'02–M'07) received B.S. degrees in computer engineering and electrical engineering from Iowa State University (ISU), Ames, in 2005. He is working towards the M.S. degree in computer engineering at ISU.

He is employed as a Software Engineer by Garmin International, Olathe, KS. He was a Teaching Assistant for CPRE 211 and CPRE 488 in the Department of Electrical and Computer Engineering at ISU in 2005 and 2006. His research interests include hardware/software codesign of embedded computer systems and embedded computer systems education.

Mr. Helvick received an ISU Teaching Excellence Award in 2007. He is a member of the Golden Key and Eta Kappa Nu Honor Societies.