Integrating Computer Engineering Lab Using Spiral Model

Dr. Pong P. Chu, Cleveland State University

Dr. Chu is Associate Professor in the Department of Electrical Engineering and Computer Science. He has taught undergraduate and graduate digital systems and computer architecture courses for more than two decades, and he has received multiple instructional grants from the National Science Foundation and authored six textbooks in this area.

Dr. Chansu Yu, Cleveland State University

Chansu Yu received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Korea, in 1982 and 1984, respectively, and the Ph.D. degree in computer engineering from the Pennsylvania State University in 1994. He is currently Chair and Professor of the Department of Electrical Engineering and Computer Science at the Cleveland State University in Cleveland, Ohio. Before joining the CSU, he was on the research staff at LG Electronics, Inc. He has authored/coauthored more than 120 technical papers and numerous book chapters in the areas of mobile computing, performance evaluation, and parallel systems. His research has been supported by both industry and government including National Science Foundation. Dr. Yu is a member of the IEEE, IEEE Computer Society, ACM, and ASEE.

Dr. Karla R Hamlen, Cleveland State University

Dr. Karla Hamlen is an Associate Professor of Educational Research in the Department of Curriculum and Foundations. She specializes in educational research relating to both formal and informal entertainment technology use among students.

Integrating Computer Engineering Lab Using Spiral Model

1. Introduction

1.1 Motivation

Recent engineering education studies call for change to enhance student learning and to better prepare graduates to meet the new challenge[1,2,3]. A good engineer should have a deep understanding of a domain and can apply the knowledge to solve problems[4]. This requires two types of practices – the "component skill," which is the knowledge of a specific domain, and the "integration skill," which applies and integrates component skill to address complex and realistic problems[5]. The Carnegie Foundation for the Advancement of Teaching conducted a five-year study of engineering education and published the results in a book titled *Educating Engineer: Designing for the Future of the Field* [3]. It points out that one deficiency is that engineering curricula mainly focus on the component skill and teach each subject in isolation and without proper context. Students are not adequately prepared for the integration skill. The study recommends a "spiral model" to provide more effective learning experiences:

> "*… the ideal learning trajectory is a spiral, with all components revisited at increasing levels of sophistication and interconnection. Learning in one area supports learning in another*."

The study also calls the labs a missed opportunity and states that[3]:

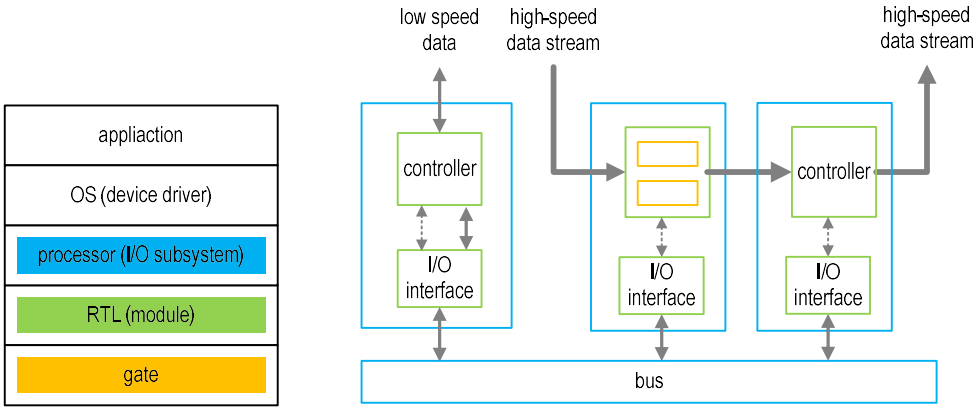
> *"…[The labs] can be more effectively used in the curriculum to support integration and synthesis of knowledge, development of persistence, skills in formulating and solving problems, and skills of collaboration. Design projects offer opportunities to approximate professional practice, with its concerns for social implications; integrate and synthesize knowledge; and develop skills of persistence, creativity, and teamwork."*

Our work is motivated by the study and develops a continuous and cohesive computer engineering lab framework based on the recommended spiral model.

1.2 "Disconnection" in computer engineering curriculum

Abstraction is used in computer engineering and computer science to manage complexity. A system is depicted by a layered model. A simple model is shown in Figure 1(a). The computer engineering curriculum is basically aligned with the abstraction layers of the model. A course usually focuses on one layer and provides detailed coverage. These courses are taught as independent topics and become isolated and disconnected pieces. While students learn the specific details of each layer, they do not have a comprehension of the relationships among the layers. This leads to the lack of integration skill.

Our work connects and integrates the individual courses through a cohesive lab framework. The lab framework will keep the lecture content intact but update the experiments and projects to make students aware of the big picture, help them to relate the individual subjects, and apply and integrate the previous learning in a new context.



(a) abstraction layer (b) IP core model

Figure 1. Layered IP core model

1.3 Theme based lab framework

Our work establishes a framework with *three themes*, which are:

- Video (image)
- Sound
- Touch sensor

Each theme consists of an array of components and an IP (intellectual property) core will be developed for each component. The complexity of the component and the abstraction levels of the design increase gradually as students progress through the curriculum. Each design starts at the gate level and eventually evolves into a customized IP core. These IP cores constitute an I/O subsystem and a video subsystem that can be integrated into any FPGA (field programmable gate array) based embedded computer system.

These themes are spread over the entire curriculum and woven together from a freshman introduction engineering course to a senior capstone design course. The key concepts are repeated in different courses with increasing complexity and sophistication and studied from different aspects and contexts, such as the software implementation versus the hardware implementation, the gate-level design versus the system-level integration, etc.

2. System Development

Our work establishes a framework with three themes for the entire computer engineering curriculum. Each theme grows in two dimensions:

- Component complexity.
- Design abstraction level.

The component dimension represents the I/O devices and peripherals. Each theme uses an array of I/O components and modules. While all the components in a theme follow the same basic operation principle and perform similar functions, their capabilities and complexities gradually grow. The design abstraction dimension reflects the abstraction layers of an IP core shown in Figure 1(a). The construction starts at the gate level or register-transfer level and gradually evolves to an IP core with software driver library.

2.1 Components in each theme

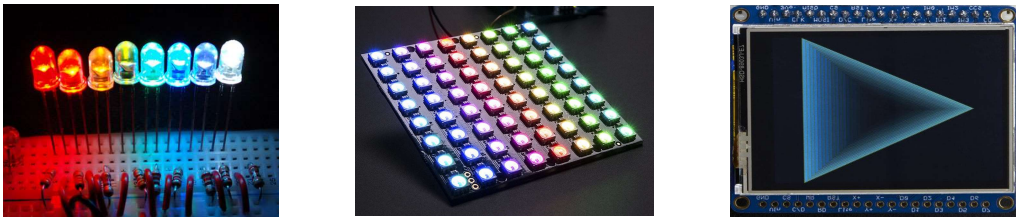
2.1.1 Video (image) theme

A computer image is composed of a matrix of *pixels*. A pixel contains three primary colors (red, green, and blue) and different colors can be obtained by adjusting the intensities of the primary colors. For example, in the VGA mode[6], an image is composed of a 640-by-480 matrix (i.e., 640 pixels in a row and 480 pixels in a column) and has about 300 thousand (i.e., 640*480) pixels.

The video theme involves the display of an image. The components in the theme gradually increase the number of pixels within the display. These devices and modules are

- One-pixel device: a tricolor discrete LED for display.
- 64-pixel device: an 8-by-8 WS2812 based LED matrix[7], which represents a display with 64 pixels.
- Low-resolution device: a 320-by-240 TFT (thin film transistor) module[8], which contains a TFT LCD display and an integrated controller.
- High-resolution device: 640-by-480 VGA display.

The first three devices are shown in Figure 2.



(a) Discrete LEDs (b) LED matrix (c) TFT display

Figure 2. Video theme devices

2.1.2 Sound theme

The sound theme involves the generation of a sound signal of specific frequency and "shape." The components in the theme concern the digital circuits that generate the signal. The complexity increases with the desired shape of the waveform:

- Square wave
- Sinusoidal wave
- Modulated sinusoidal wave

The theme leads to a DDFS (direct digital frequency synthesis) core and an ADSR (attack-decay-sustain-release) envelope generator[9,10,11].

2.1.3 Touch sensor theme

The touch sensor theme involves the methods to obtain an input from buttons or a touch force sensor. The components in theme gradually increase the number of sensor points within the device. These devices and modules are:

- One-bit devices: a push button switch and a force sensitive resister.

- Multi-bit device: a keypad, which detects the activation of a key by applying a scanning signal.
- Resistive touch screen: a touch screen, which is a thin layer of two-dimensional resister matrix overlaid on a portable device and detects the pressed location by applying scanning signals to the x- and y-axes alternatively and measuring the responses.

The devices are shown in Figure 3.

(a) Force sensitive resistor (b) Keypad (c) Touch screen

Figure 3. Touch sensor theme devices

2.2 IP core architecture

An IP core is developed for each component. The core is created from scratch. Its construction follows the layered model in Figure 1(a) and is completed in a bottom-up fashion. In the gate layer, the circuits are constructed with small and medium sized logic components. In the RTL layer, the circuits are assembled to form a larger module. In the processor layer, the module is augmented with additional decoding circuit and buffering registers to interface with the system bus and to communicate with the processor. The three layers and their corresponding parts are depicted in Figure 1(b). In the OS layer, the software driver routines are derived to access the core.

There are two basic architectures. The block diagram of an *I/O IP core* is shown in the left of Figure 1(b). It consists of a controller module, which accesses and controls an external device, and a bus interface module, which communicates with the processor. The block diagram of a *video IP core* is shown in the right of Figure 1(b). It can be treated as a hardware accelerator with separated data flow and control flow. The high-speed data stream is processed with customized computation modules and routed through the dedicated paths. The control is connected via a similar bus interface module.

The following PWM (pule width modulation) IP core example demonstrates the development procedure. The PWM IP core sets the color of a tricolor LED. A PWM controller is constructed with a binary counter and a comparator, as shown in Figure 4. The counter runs and wraps around continuously. Its output is compared with the `duty_cycle` signal and the asserted accordingly. The PWM IP core is composed of three PWM controllers and a bus interface circuit and its block diagram is shown in Figure 5. The processor can set the PWM duty cycle by writing the designated register.

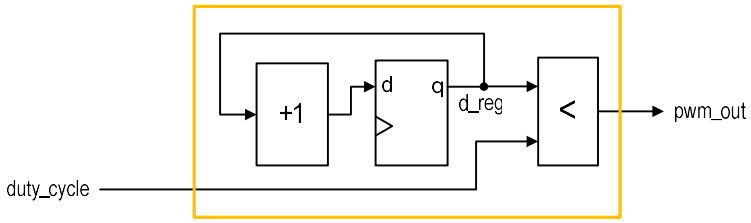


Figure 4. PWM controller

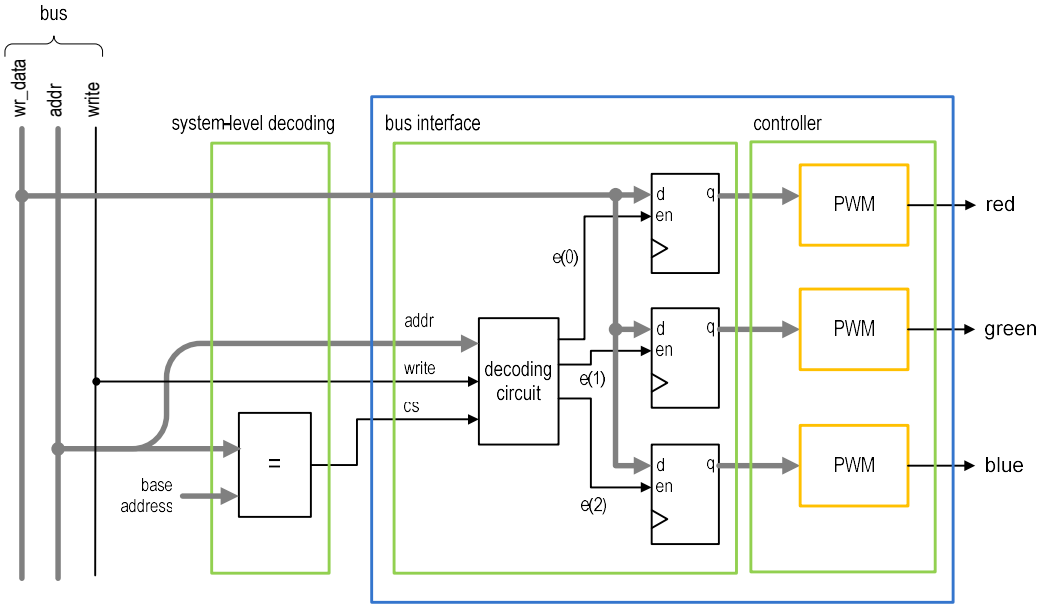


Figure 5. Three-channel PWM IP core

2.3 System architecture

Our work develops six video IP cores, two sound IP cores, and a collection of general-purpose I/O IP cores that support various I/O interfaces and devices. The detailed design and coding of these cores are documented in two texts[12,13].

To achieve maximal portability, we define a simple uniform bus protocol for all IP cores. Since the high-speed data stream is routed separately, the bus is only used to configure the core, get status, issue commands, and transmit and receive low-speed data. The bus interface circuit just involves decoding and multiplexing circuits and buffering registers, as shown in Figure 5. The FPGA bus protocols, such as the Avalon bus and the AXI bus, are very complicated and are not compatible with each other. We use a *bridge* to translate the signals between an FPGA's native bus protocol and our proposed protocol.

The block diagram of the complete system is shown in Figure 6. Our work develops a collection of IP cores for the I/O and video subsystems but does not "reinvent" the processor. It utilizes an existing processor, such as MicroBlaze from Xilinx or Nios II from Altera, and

incorporates the IP cores to form a complete system. The shaded portion of Figure 6 are the vendor's cores.

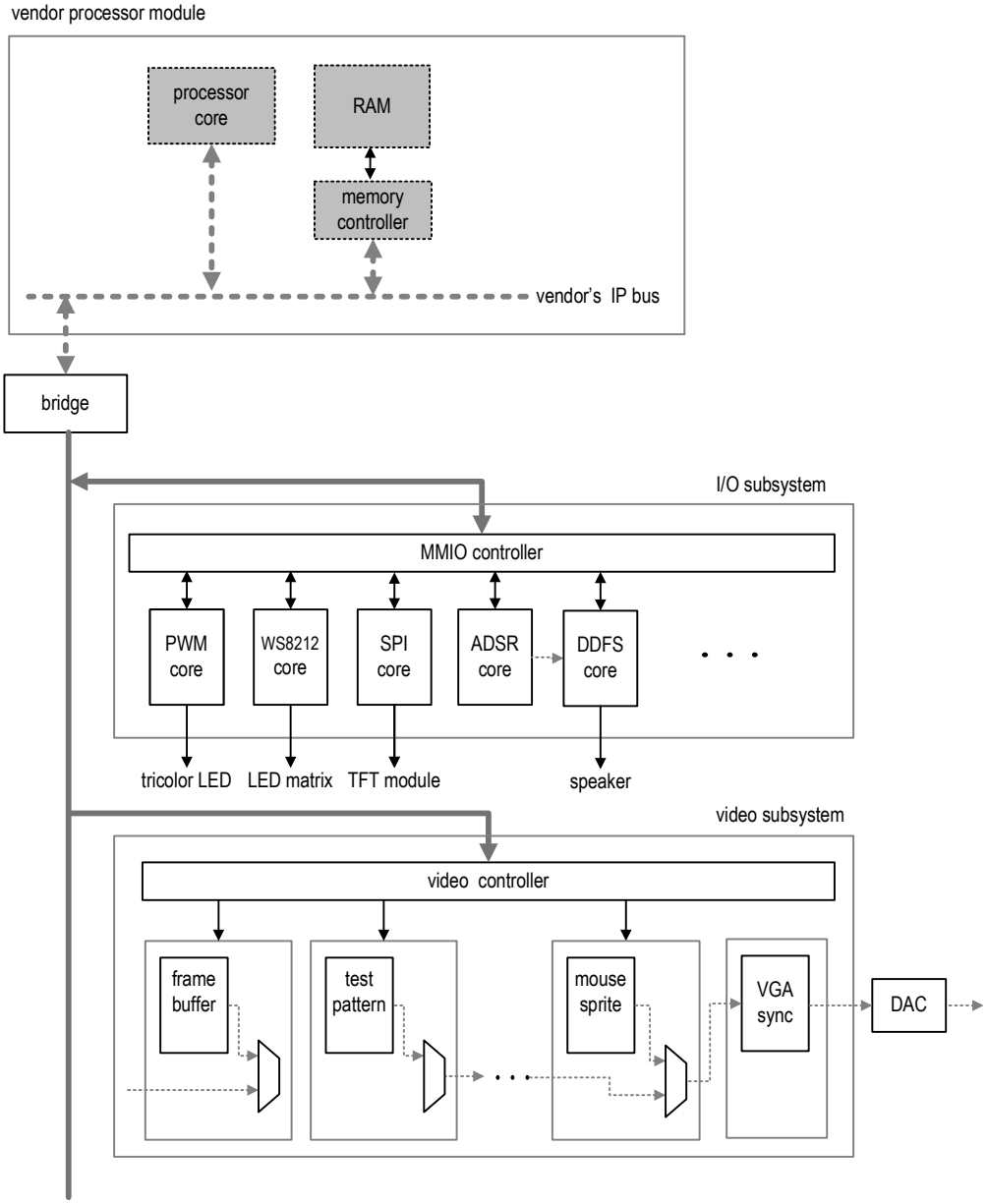


Figure 6. Complete system

2.4 Prototyping boards

Our work uses a simple microcontroller board and an entry-level FPGA prototyping board. The former is used for the introductory freshman lab and the latter is used for the remaining labs.

We select the Arduino Uno board as the microcontroller board[14]. It contains an 8-bit microcontroller and a dozen connectors to access external I/O devices. It supports a subset of C/C++ and provides a simple user development environment. The setup is targeted for beginners without much prior programming or hardware experience.

An FPGA device is a semiconductor device that contains a collection of generic logic elements and interconnects. The logic elements and interconnects can be configured to perform a specific function. An FPGA prototyping board typically contains an FPGA chip, a memory chip, various I/O peripherals, and auxiliary programming and power circuits[15,16]. The experiments and projects are implemented and tested on the Digilent Nexys 4 DDR board. However, since our designs and HDL codes are portable and not tied to a specific vendor, they can be easily migrated to other boards.

3. Lab Experiments and Projects

We divide the lab experiments and lab projects into four levels:

- Level 1: Freshman engineering.
- Level 2: Basic digital system.
- Level 3: Advanced digital system without a processor.
- Level 4: Advanced digital system with a processor.
- Level 5: Capstone projects.

The level 1 is for freshman engineering students. Many schools now have an "introduction to engineering'' course for the new engineering students. It is usually a project-oriented course to introduce the basic engineering concepts and practices.

The level 2 corresponds to the first digital system course in the curriculum, which covers the combinational circuits, sequential circuits, and FSM[17].

Unlike the first digital system course, there is no single "standard" follow-up course. The advanced topics can be spread over a wide variety of courses, such as advanced digital system, computer organization, VHDL/Verilog, embedded system, hardware-software codesign, and so on. For our development purposes, one key distinction is whether a processor is incorporated into the course. Based on this, we divide the follow-up into two levels – level 3 (advanced digital system without a processor) and level 4 (advanced digital system with a processor). The latter implies that we go further on the layered model to include the processor and software drivers in a design. The complete system and the detailed design of IP cores are covered when students finish the advanced computer engineering courses. The level 5 is to apply the materials from the previous levels for a multi-week term project or a capstone design project.

The suggested experiments and projects in the sound theme and the video theme are listed and detailed in separate articles[18,19,20] and are not repeated here. We just demonstrate an example sequence on the video theme in this section. A "rainbow strip" displays a continuous rainbow-like light spectrum[21], as shown in Figure 7. It is obtained by gradually incrementing or decrementing the intensity of the red, green, or blue color.

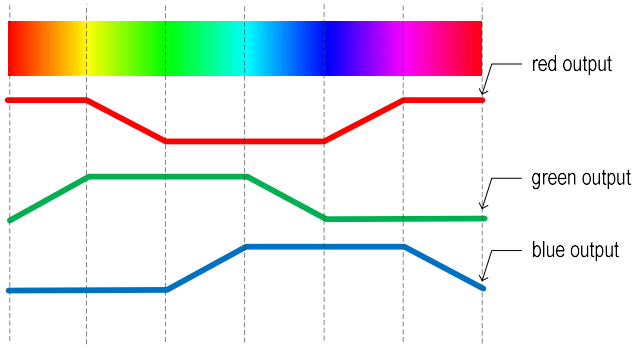


Figure 6. Rainbow spectrum

Following are the experiments in various levels:

1. Rainbow nightlight with Arduino (level 1). We can use a tricolor LED as a nightlight. It gradually circulates the colors of the rainbow spectrum of Figure 7 and then wraps around. The experiment can be done with Arduino's built-in `analogWrite()` function, which uses processor's PWM module.
2. TFT Rainbow with Arduino (level 1). The TFT module contains a frame buffer, which can be accessed via the SPI bus. The rainbow can be displayed on the screen by using the pre-designed library function to write the desired colors to the frame buffer.
3. PWM controller (level 2). This constructs the PWM controller of Figure 4.
4. Rainbow nightlight with customized hardware (level 2). This repeats the Experiment 1 but uses pure hardware implementation. The design includes three PWM controllers and a top-level master timing circuit.
5. SPI controller (level 3). This designs an SPI controller to communicate with the TFT module.
6. TFT rainbow with customized hardware (level 3). This repeats the Experiment 2 but uses pure hardware implementation. A master control circuit performs the TFT initialization and writes the frame buffer.
7. VGA rainbow test pattern generator (level 3). This generates a rainbow test pattern on a VGA monitor.
8. PWM IP core and driver (level 4). This develops the multi-channel PWM IP core of Figure 5 and derives the driver library. The experiment 1 of Level 1 can be repeated with the core and the driver library.
9. SPI IP core and TFT driver (level 4). This develops the SPI IP core and derives the TFT driver library. The Experiment 2 can be repeated with the core and the driver library.

4. Integration to Existing Curricula

Our work develops several cohesive series of theme-based experiments and projects. The experiments are intended for the companion hands-on exercises and projects of the theoretical topics covered in the lecture portion of a course. Each experiment is based on a specific theoretical subject area and can be conducted as an independent exercise. For example, in the subject area of a sequential circuit, the lecture portion will cover the theory and the principle of a basic counter. The PWM controller can be used as a practical application example for the

subject area. The experiment also contains additional "integration exercises" that relate to the `analogWrite()` function of the previous level and served as a basis for the multi-channel PWM IP core of the next level. While understating the previous experiments in other levels and doing integration exercises are beneficial, they can be omitted and the PWM controller can be used as a stand-alone experiment.

Adopting and integrating the experiments into an existing curriculum is straightforward and flexible. An instructor first identifies the corresponding experiment of a specific subject area and then substitutes it with a theme based experiment in the proposed project. Depending on the nature of the course, the instructor can replace as many experiments as is appropriate. There is no need to modify the existing course structure since only the lab portion is updated. To achieve the best result, the instructors in charge of various courses should agree on a theme and select the experiments from that theme accordingly.

The estimated cost of materials of a single setup is about $250, including a microprocessor board ($25), an FPGA board ($150), and various components ($75). Most curricula already use an FPGA board in lab. Since the developed experiment codes are portable and device independent, the existing board can be used and thus adoption requires no significant amount of funds.

5. Evaluation

The effectiveness of the spiral curriculum in this study is evaluated through the use of a quasi-experimental design. The goal of the evaluation is to determine whether the revised curriculum is more effective for student learning, student interest, and whether students and instructors feel it is a more effective method as well as the adequacy of lab materials. The control group for the evaluation consists of students taking courses in current curriculum and the experimental group consists of the students taking courses in the enhanced spiral curriculum. The evaluation plan is a mixed-methods approach consisting of a quasi-experimental design with additional qualitative data to support findings. A student interest and opinion survey was given at the beginning and end of the semester, in which they reported their level of agreement with several items. The survey given in this study was based off the Science Motivation Questionnaire II, an instrument which has been validated for assessing students' motivation to learn science in college[22]. Questions were modified to be relevant to engineering and computer science, and additional questions were added that were specific to the labs since they are an essential component of the revised curriculum in this study.

Our work is scheduled to be completed in the next year and the data is still being collected for the study. The preliminary descriptive statistics are provided in this section and the inferential analyses will be reported later. Thus far we have 129 survey responses. Since some students took the survey multiple times because they took a sequence of courses all using the revised curriculum, for preliminary results we will only give the results from the first time a student experienced the new curriculum. We have data from 74 unique students from two universities who experienced the revised curriculum for the first time in one of their classes. Of these, 87.8% of the sample was male, and 12.2% was female. 19% self-reported as Asian, 8% as Black or African American, 46% White, and 8% Middle Eastern or Arab. The mean age of students taking this survey was 24 (with a standard deviation of 5.34).

Each survey item was scored from 1 through 5, with 1 being strongly disagree and 5 being strongly agree. After taking their first course with the revised curriculum, mean student responses to the questions focused on the labs were summarized in Table 1.

Question	Mean	Sd
The lab/project work I do for this course is relevant to my learning	4.20	0.71
Doing the labs/projects for this course is interesting for me	4.40	0.72
The labs/projects for this course show me how to problem-solve in Computer Engineering	4.30	0.64
The labs/projects in this course make the content more understandable	4.20	0.84
Understanding the content of this course will benefit me in my career	4.10	0.75
Doing the labs/projects shows me real life applications of the information	4.10	0.75

Table 1. Survey question result

Open ended questions at the end of the survey allowed respondents to expand on their answers and discuss the role labs played in their learning. Most respondents responded that the labs were extremely helpful in their learning, and several even noted that they were the most helpful part of the course since they like to learn using hands-on methods and the labs allowed them to see real-life connections to the course content. There was also a common sentiment that the labs required too many hours and were too difficult. Based on this information, some modifications will be made to the curriculum for the next groups of students experiencing the revised curriculum.

6. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1504030. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. Photos in Figures 2 and 3 are courtesy of Adafruit.com.

Bibliography

[1]. S. A. Ambrose et al., *How Learning Works: Seven Research-Based Principles for Smart Teaching*. Jossey-Bass, 2010.

[2]. C. J. Atman, et al., *Enabling Engineering Student Success: The Final Report for the Center for the Advancement of Engineering Education*, 2010.

[3]. S. Sheppard, et al., *Educating Engineers: Designing for the Future of the Field*. Jossey-Bass, 2009.

[4]. T. A. Litzinger, et al., "Engineering Education and the Development of Expertise," *Journal of Engineering Education*, pp.123-150, January 2011.

[5]. J. E. Froyd and M. W. Ohland, "Integrated Engineering Curricula," *Journal of Engineering Education*, pp.147-164, January 2005.
[6]. Wikipedia web site https://en.wikipedia.org/wiki/Video_Graphics_Array.
[7]. WorldSemi, *WS2812 intelligent control LED integrated light source datasheet*.
[8]. ILITEK, *ILI9341 TFT LCD Single Chip Driver Specification*.
[9]. M. Genovese, et al., "Analysis and comparison of Direct Digital Frequency Synthesizers implemented on FPGA," *Integr. VLSI Journal*, March 2014.
[10]. J. Vankka, *Direct Digital Synthesizers: Theory, Design and Applications*, Springer, 2001.
[11]. Wikipedia web site https://en.wikipedia.org/wiki/Synthesizer#ADSR_envelope.
[12]. P. Chu, *FPGA prototyping by VHDL examples, 2nd edition*, John Wiley & Sons, Inc., 2017.
[13]. P. Chu, *FPGA prototyping by Verilog examples, 2nd edition*, John Wiley & Sons, Inc., to be published in 2018.
[14]. S. Monk, *Programming Arduino: Getting Started with Sketches*, McGraw-Hill, 2011.
[15]. Terasic web site http://www.terasic.com.tw/en/.
[16]. Digilent web site https://store.digilentinc.com/fpga-programmable-logic/.
[17]. ACM/IEEE Computer Society, *Computer Engineering Curricula*, 2016.
[18]. Pong P. Chu, Chansu Yu, and Karla Mansour, "A Spiral Computer Engineering Lab Framework," accepted for presentation *in AAAS/NSF Symposium on Envisioning the Future of Undergraduate STEM Education: Research and Practice*, 2016.
[19]. P. Chu, "Integrating Computer Engineering Labs with a Sound Theme," *ASEE (American Society for Engineering Education) Annual Conference*, 2016.
[20]. P. Chu, "Integrating Computer Engineering Labs with a Video Theme," *ASEE (American Society for Engineering Education) Annual Conference*, 2017.
[21]. Wikipedia web site https://en.wikipedia.org/wiki/HSL_and_HSV.
[22]. S. M. Glynn et.al., 011 "Science Motivation Questionnaire II: Validation with science majors and nonscience majors," *Journal of Research in Science Teaching*, 48(10), p.1159-1176, 2011.