Software-hardware Integration of System Design Discipline

Dr. Wangling Yu, Purdue University Northwest

Dr. Wangling Yu is an assistant professor in the Electrical & Computer Engineering Technology Department of the Purdue University Northwest. He was a test engineer over 15 years, providing technical leadership in the certification, testing and evaluation of custom integrated security systems. He received his PhD degree in Electrical Engineering from the City University of New York in 1992, specializing in control theory and electronic technology.

Prof. Omer Farook, Purdue University Northwest

Omer Farook is a member of the faculty of Electrical and Computer Engineering Technology at Purdue University, Nothwest. Farook received the diploma of licentiate in mechanical engineering and B.S.M.E. in 1970 and 1972, respectively. He further received B.S.E.E. and M.S.E.E. in 1978 and 1983, respectively, from Illinois Institute of Technology. Farook's current interests are in the areas of embedded system design, hardware-software interfacing, digital communication, networking, image processing, and biometrics, C++, Python, PHP and Java languages. He has a keen interest in pedagogy and instruction delivery methods related to distance learning. He has a deep commitment to social justice and in achieving economic and educational equity.

Dr. Jai P. Agrawal, Purdue University Northwest

Jai P. Agrawal is a professor in electrical and computer engineering technology at Purdue University, Calumet. He received his Ph.D. in electrical engineering from University of Illinois, Chicago, in 1991, dissertation in power electronics. He also received M.S. and B.S. degrees in electrical engineering from Indian Institute of Technology, Kanpur, India, in 1970 and 1968, respectively. His expertise includes analog and digital electronics design, power electronics, nanophotonics, and optical/wireless networking systems. He has designed several models of high frequency oscilloscopes and other electronic test and measuring instruments as an entrepreneur. He has delivered invited short courses in Penang, Malaysia and Singapore. He is also the author of a textbook in power electronics, published by Prentice-Hall, Inc. His professional career is equally divided in academia and industry. He has authored several research papers in IEEE journals and conferences. His current research is focused on renewable energy technology, smart energy grid.

Prof. Ashfaq Ahmed P.E., Purdue University Northwest

Ashfaq Ahmed is a Professor of Electrical and Computer Engineering Technology at Purdue University Northwest. Ahmed received his bachelor's of science degree in electrical engineering from the University of Karachi in 1973 and master's of applied science degree in 1978 from University of Waterloo. He is the author of a textbook on power electronics, published by Prentice-Hall. He is a registered Professional Engineer in the state of Indiana. He is a senior member of IEEE. Ahmed's current interests include embedded system design, electric vehicle, and VHDL design.

©American Society for Engineering Education, 2018

Software Hardware Integration of System Design Discipline in Electrical and Computer Engineering Technology

Abstract

The paper expounds the practices utilized in teaching an undergraduate curriculum in Electrical and Computer Engineering / Technology from the perspective of System Design. This approach is a paradigm shift from the piecemeal methodology of dividing the discipline into bifurcated courses. Rather, this approach prepares the student to pursue the discipline of System Design from at least four different perspectives, via: 1) PLD/FPGA centric system design, 2) Microcontroller-based Embedded System Design, 3) PC based / or System on the Chip (SoC)-based systems such as Beagle Bone Black-based Network-oriented Distributed System Design, IoT and cloud-centered programming, and 4) DSP-based Real-time Processing-Based System Design. The paper will summarize the content of eight total courses that all have the common vein of system design.

Also examined in this paper is the origin of Outcome Based Education (OBE) as a philosophy and its implementation in our curriculum, designed and delivered with the principles of Outcome Based Education. As such, it lends itself to an Outcome Based Assessment, which is the cornerstone of ABET. The paper presents details of the protocols that were utilized and adhered to in the implementation of OBE.

Furthermore, the paper also discusses a set of courses in the areas of hardware, software, firmware, networking and DSP, which provides a roadmap in the form of a curriculum that utilizes the same tools the industry is employing. The content of these courses is presented to the students in the context of system design, and the courses are conducted in an environment that does not follow the traditional routine of lectures and labs, which are secluded from each other and disjointed. Instead, the total discourse of these courses takes place in a lab/studio setting, and always with reference to system being designed. Such an approach bridges the gap that exists between classroom practices and workplace practices. This unified approach is meant to bring about students with career-bound knowledge that is essential for the industry.

I. Introduction

In the Electrical and Computer Engineering Technology (ECET) department of Purdue University Northwest, the faculty by choice have taken upon themselves to make the aforementioned changes in their curricula, the essence of which is reflected in this paper. We the faculty here are revisiting the subject matter after a 15-year time span. We have kept up with the changes in technology during this timeframe and have made myriad corresponding changes [1]. We kept the content of these courses up-to-date and incorporated a host of pedagogical experimentations that resulted in better classroom learning. The curriculum now presented here prepares the student to function in the marketplace as a System Designer. It was realized that the ECET curriculum should impart to the graduating student enough exposure to pursue the discipline of System Design from at least four different perspectives, via: 1) PLD/FPGA centric system design, 2) Microcontroller-based Embedded System Design, 3) PC based / or System on the Chip (SoC)-based systems such as Beagle Bone Black-based Network-oriented Distributed System Design, IoT and cloud-centered programming, and 4) DSP-based Real-time Processing based System Design. The Industrial Advisory committee was fully supportive in this process, providing constructive suggestions in this regard. It was with this objective that a curriculum overhaul was done, and the sequences of two courses at the minimum were slated in each of the four areas. Thus, a total of eight courses were designed and are being offered over a four-year time span, spread among eight semesters of a bachelor's degree program in ECET.

II. PLD Based System Design

Digital Logic Design Courses in undergraduate ECET programs have undergone a long evolution. A traditional two-course sequence in Combinational Logic Circuit and Sequential Logic Circuit is taught using Fixed Function Logic with TTL or CMOS devices [2]. In our department, these courses were redefined, so that they are now taught using Programmable Logic Devices (PLD's) [3]. We have included Field Programmable Gate Array (FPGA) as the fundamental component of the laboratory experience. This change was necessary to introduce students to the practices of modern industry that utilizes FPGA for digital design.

Programmable Logic Devices are cost-effective and give hardware designers the flexibility of an ASIC with very short turnaround time. PLDs give the designer the ability to make changes easily during the development phase through simple changes in the program without making expensive, time-consuming changes to hardware and wiring connections. Finally, PLDs are easy to work with as compared to Fixed Function Logic, where students often make errors in connections even for a simple logic circuit.

The first course in the sequence introduces design entry using Schematic Capture. The students can relate to the logic circuit as learned in the classroom using traditional logic symbols and Boolean algebra principles. This is followed by some functional simulation practice to verify the basic design. The final step is then performed by downloading the programming file onto the target FPGA device. In the next phase, the Very High Speed Hardware Description Language (VHDL) design styles are introduced. The students are given a taste of the three approaches to writing Architecture Declaration, namely: (a) Data-flow Approach (b) Structural Approach and (c) Behavioral Approach. With this background, the students become proficient in System Design using FPGA.

The first course in the sequence is Digital Fundamentals [ECET 109, Lecture 2: Lab. 2: Credit 3] and it covers the following topics:

- Basic Principles of Digital Systems, Logic functions and gates, Boolean algebra, and Combinational Logic.
- Introduction to PLDs and Quartus Prime software by Intel -Altera Corporation.
- Programming PLDs using Quartus Prime.
- Designing simple combinational circuits using schematic capture.

- Using Very High Speed Integrated Circuit (VHSIC): VHSIC Hardware Description Language (VHDL) declarations, architecture and concurrent signal assignments to enter simple combinational circuits.
- Creating circuit symbols from schematics or VHDL designs for PLDs.
- Pursuing various Combinational Logic functions and arithmetic circuits through PLDs and VHDL programming.

The second course in this sequence is Digital Applications [ECET 159, 2:3:2] beginning with introduction to Sequential Logic. Then the following topics are covered:

- Design applications using NAND/NOR Latches, Gated Latches, D, JK and T Flip-Flops.
- Programmable Logic Architecture of devices (Cyclone IV E FPGA EP4CE115F29C7) is explored.
- Counters and Shift Register design is implemented using Quartus Prime Graphic Design File or in VHDL.
- The course culminates with the “classic” (state table) method of the State Machine Design. This leads to the implementation of the State Machine using state diagram.
- Finally write VHDL code to implement the State Machine Design.

A short list of some typical labs:

1	Combinational Logic Circuits
2	Logic Circuit Design
3	Design of a 15¢ Pencil Machine
4	Introduction to VHDL
5	VHDL using Behavioral Description
6	Binary Adders
7	Decoders and Encoders
8	Seven Segment Display
9	Multiplexers and Demultiplexers
10	Comparators
11	D and JK Flip Flops
12	Storage and Display System
13	Asynchronous Counters
14	Synchronous Counters
15	Synchronous Counter Design
16	Electronic Key
17	Pattern Detector
18	State Machine Designs
19	Shift Register
20	Multivibrators
21	Digital to Analog Converters
22	Analog to Digital Converters
23	Random Access Memory
24	Read-Only Memory

III. Microcontroller-Based Embedded System Design

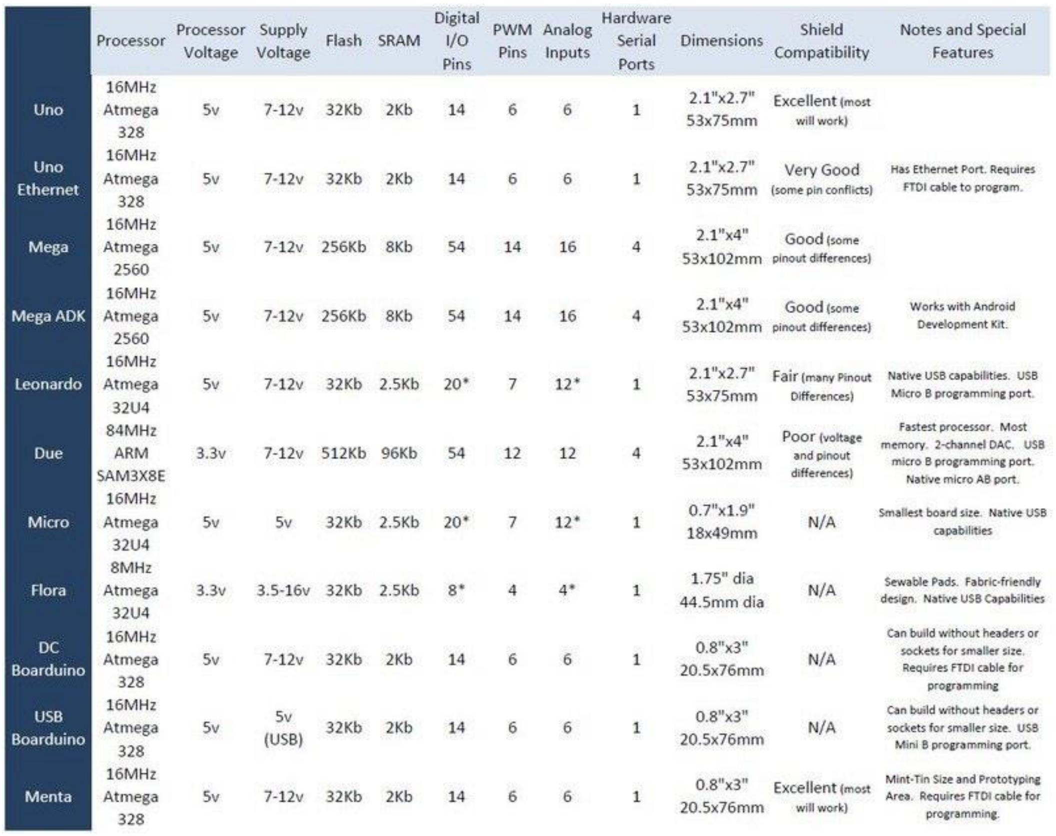
Most of the ECET programs have at least one introductory course in either Microprocessors or Microcontrollers. In days past, such a course was typically taught in Assembly language. Our department was the one of the early pioneers to make a shift from Assembly language to using C language. Over a period spanning more than two decades, we engaged in objective self-analysis and made a number of continually corrective improvements to keep the course current. This evolutionary process took us from 8085, to 8088 microprocessors, to 8051 microcontroller, to Microchip[4] PIC 16F84, to 16F877 microcontroller, to the presently used Atmel ATmega328P, using the open platform of Arduino Uno Rev 3 [5]. The course is presently based in application design using C language. This course is preceded by an introductory course in C++, ECET 21000, [03:03:04] where students learn Structured Programing and learn the use of pointers for inter-functional communication. This practice is still adhered to in the present course of Microcontroller based System Design, ECET 20900, [03:03:04].

Microcontroller-based System Design, ECET 20900, starts with architectural details of Atmel ATmega328P, and students learn to use a data sheet [6]. The course uses the textbook as reference material. [7] For most of the simple systems' design, Microcontrollers are inherently better suited; for the beginning student, the Microcontroller route provides all the functionality in a neat package.

Arduino Uno is the most popular Arduino platform in the family of the Arduino product line. The following table (Figure No. 1) compares the basic features of the various Arduinos and Arduino Compatibles platforms presently available. The user has a choice among the many Arduino platforms with regard to 1) a processor and its speed, 2) Physical footprint, 3) Number of I/Os, 4) Memory size, 5) Compatibility with the daughter boards ("Shield" in Arduino terminology). A very important consideration to note is that the user has a large list of daughter boards to choose from in order to further define the user's considerations.

Arduino Uno for our considerations is typically suitable due to its low cost and its versatility for classroom use. Arduino Uno is based on ATmega 328P processor, belonging to AVR family of microcontrollers developed by Atmel. These are modified Harvard architecture 8-bit RISC single-chip microcontrollers. AVR was one of the first microcontroller families to use on-chip flash memory for program storage. Arduino Uno is available to operate at 16 MHz out of the box. The user has three memory pools to choose from: 1) Flash memory (program space), where the Arduino stores its application program (sketch in Arduino terminology). 2) SRAM (static random access memory) is where the application program creates and operates variables when it executes. 3) EEPROM is where long-term information can be stored.

Flash memory and EEPROM memory are non-volatile (the information persists after the power is turned off). SRAM is volatile and will be lost when the power is turned off. The ATmega 328 chip found on the Uno has the following amounts of memory: 1) Flash 32k bytes (of which 0.5k is used for the bootloader) 2) SRAM 2k bytes 3) EEPROM 1k byte.



	Processor	Processor Voltage	Supply Voltage	Flash	SRAM	Digital I/O Pins	PWM Pins	Analog Inputs	Hardware Serial Ports	Dimensions	Shield Compatibility	Notes and Special Features
Uno	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	2.1"x2.7" 53x75mm	Excellent (most will work)	
Uno Ethernet	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	2.1"x2.7" 53x75mm	Very Good (some pin conflicts)	Has Ethernet Port. Requires FTDI cable to program.
Mega	16MHz Atmega 2560	5v	7-12v	256Kb	8Kb	54	14	16	4	2.1"x4" 53x102mm	Good (some pinout differences)	
Mega ADK	16MHz Atmega 2560	5v	7-12v	256Kb	8Kb	54	14	16	4	2.1"x4" 53x102mm	Good (some pinout differences)	Works with Android Development Kit.
Leonardo	16MHz Atmega 32U4	5v	7-12v	32Kb	2.5Kb	20*	7	12*	1	2.1"x2.7" 53x75mm	Fair (many Pinout Differences)	Native USB capabilities. USB Micro B programming port.
Due	84MHz ARM SAM3X8E	3.3v	7-12v	512Kb	96Kb	54	12	12	4	2.1"x4" 53x102mm	Poor (voltage and pinout differences)	Fastest processor. Most memory. 2-channel DAC. USB micro B programming port. Native micro AB port.
Micro	16MHz Atmega 32U4	5v	5v	32Kb	2.5Kb	20*	7	12*	1	0.7"x1.9" 18x49mm	N/A	Smallest board size. Native USB capabilities
Flora	8MHz Atmega 32U4	3.3v	3.5-16v	32Kb	2.5Kb	8*	4	4*	1	1.75" dia 44.5mm dia	N/A	Sewable Pads. Fabric-friendly design. Native USB Capabilities
DC Boarduino	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	0.8"x3" 20.5x76mm	N/A	Can build without headers or sockets for smaller size. Requires FTDI cable for programming
USB Boarduino	16MHz Atmega 328	5v	5v (USB)	32Kb	2Kb	14	6	6	1	0.8"x3" 20.5x76mm	N/A	Can build without headers or sockets for smaller size. USB Mini B programming port.
Menta	16MHz Atmega 328	5v	7-12v	32Kb	2Kb	14	6	6	1	0.8"x3" 20.5x76mm	Excellent (most will work)	Mint-Tin Size and Prototyping Area. Requires FTDI cable for programming.

Figure No. 1: Arduino Comparison Chart [8]

SparkFun's Inventors Kit [9]

The class utilized SparkFun's Inventors Kit. The kit was complete with the required set of parts and SparkFun provides a set of following 16 experiments:

Introduction: SIK RedBoard & SparkFun Mini Inventor's Kit
Introduction: SIK Arduino Uno
Experiment 1: Blinking an LED
Experiment 2: Reading a Potentiometer
Experiment 3: Driving an RGB LED
Experiment 4: Driving Multiple LEDs
Experiment 5: Push Buttons
Experiment 6: Reading a Photoresistor
Experiment 7: Reading a Temperature Sensor
Experiment 8: Driving a Servo Motor
Experiment 9: Using a Flex Sensor
Experiment 10: Reading a Soft Potentiometer

Experiment 11: Using a Piezo Buzzer
Experiment 12: Driving a Motor
Experiment 13: Using Relays
Experiment 14: Using a Shift Register
Experiment 15: Using an LCD
Experiment 16: Simon Says

The kit provides the following set of parts:

- SparkFun RedBoard
- Arduino and Breadboard Holder
- SparkFun Inventor's Kit Guidebook
- Translucent Red Bread Board
- Carrying Case
- 16x2 White on Black LCD (with headers)
- 74HC595 Shift Register
- 2N2222 Transistors
- 1N4148 Diodes
- DC Motor with Gear
- Small Servo
- SPDT 5V Relay
- TMP36 Temp Sensor
- Flex sensor
- Softpot
- 6' SparkFun USB Cable
- Jumper Wires
- Photocell
- Tri-color LED
- Red and Yellow LEDs
- 10K Trimpot
- Piezo Buzzer
- Big 12mm Buttons
- 330 and 10K Resistors

The class utilizes the set of experiments provided. The most important and positive aspect of these experiments is that each experiment has an extensive portion of theoretical engineering aspect of the discussion and reasoning in the form of comments. We have utilized these experiments as such, but the negative aspect of these experiments' code is they have not utilized standard C, and furthermore were not designed using Structured Programing Methodology. In this class we utilized the skillset of the prerequisite class, where students learned and mastered C^{++} code design, strictly following Structured Programing Methodology. Thus, our students design the code using standard C and following the principles of Structured Programing Methodology. All the inter-functional data communication is carried through utilizing Pointers. [10]

IV. Operating System with Embedded System Design, A Network-Oriented Distributed System Design

Previously, the System Design courses were mostly centered around a single processor. Before the era of Microcontrollers, courses were designed with single Microprocessors (80*86), EPROM chip, and 2048-bit static ram with i/o ports and a timer (8155) in a minimum mode. This era was followed by the availability of Microcontrollers, and courses were designed around a single Microcontroller with all the needed functionality (A/D, Timer, RAM, ROM, and Direct availability of multiple ports) available on a single chip. This was followed by the courses that were centered around Personal Computers and Microcontrollers providing the System's functionality together. These systems tap into the dedicated processor of Microcontrollers, utilizing the power of the PC's Operating System and other resources and code that were available. This was the era Multiprocessor System Design. Availability of NET added the possibility of Distributed System Design with Multiple Processors. Moving into present times, the System Design courses are centered around the powerful new system-on-a-chip (SOC) devices (BeagleBone Black [11] and Raspberry PI [12]) that were essentially capable of performing all the duties of a computer on a single chip. The need to go beyond the basics of providing an introductory course in the microprocessor or microcontroller in Engineering- and Engineering Technology-type curriculums has long been overdue.

Operating System with Embedded System Design, is ECET 45600, [3:3:4]. This course has retained its technological currency by climbing the evolutionary ladder of myriad of technological advances in hardware, software and OSs. In its current form, the course has been totally redesigned based on BeagleBone Black. The BeagleBone Black is a compact, affordable, open-source Linux computing platform ideal for class room learning and designing. The operating system for the board is Linux and course is delivered with C^{++} and Python.

The course starts with the architecture of the board, its multiple processors and accompanying hardware components that are utilized in the course. It will further discuss the software methodologies utilized in the course with C^{++} and Python languages. Debian Linux is utilized in the course with command line environment. Class utilize these different tools employed like Putty and WinSCP and Eclipse IDE that are needed in the execution of software Design on PC. Laboratory exercises covered the interfacing, controlling, and communicating with the physical environment. We have opted into Python and C^{++} as the languages of choice for the course. Students come to this particular course with the background of two courses: one course in C^{++} and another course in Embedded System Design with Arduino. However, for this course, Python is utilized due to its compatibility and ease of use with the Adafruit BeagleBone Python library,[13] which can handle GPIO, PWM, ADC, I2C, SPI and UART. This in a nutshell serves all of our class' experimental needs.

About ¼ of the time is dedicated to learning and mastering Python language. Students pick up Python language fairly quickly since they already have a background in C^{++}. All the coding is done using strictly Structured Programing Methodology.

IV. a. Laboratory Experiments

Another ¼ of the time is dedicated to performing the lab experiments. The Laboratory experiments span the Linux operating system, Hardware Interfacing of Electronics, data gathering, analysis and control of local elements and remote elements over the NET.

IV. b. Partial List of Experiments Performed

The following is a short list of experiments performed in the lab.

1) Introduction to BeagleBone Black
2) Cloud (Web IDE Accessing the Linux Terminal)
3) Directories, User Navigation and Blinking LEDs Basic LED Python Program
4) Reading Pushbutton States
5) Analog to Digital Conversion
6) USART Serial Communication
7) Data Manipulation Using Lists (Python) or Arrays (C^{++})
8) I2C bus interface
9) Java Runtime Environment on the BBB
10) Controlling of Stepper Motors
11) DC Motor Speed AND Direction Control
12) Interfacing with Text Files (Reading and Writing)
13) Cross-Compilation and the Eclipse IDE (Home setup, not performed in Lab)
14) Remote server Interface

The succeeding second course could take a number of different routes:

1) This course, ECET 45500, [3:3:4], Object Oriented System Design could be pursed entirely in a Windows GUI environment. Window's GUI Software System Design is an area that is in high demand in the marketplace, yet virtually shunned by the academic community. The platform for the software design is Embarcadero (Old Borland C++ Builder). This approach provides the student to program entirely in a GUI, Object Oriented Programming environment.
2) This course requires students to have a background in networking so that when the components of socket programming and connectivity are taught in this course, the course is wholly disseminated in their understanding of the system(s), and their learning is made more complete. The ECET department has offered this course (ECET 499) experimentally during the past few semesters.

The impact of the above sequence of courses could be measured by the fact that 75% of the senior design projects utilize the core knowledge gained. Operating System with Embedded System Design provides a convenient mechanism to design any customize system, regardless of end usage. This provides the student the knowledge base for Hardware, Software Integration. [14] Please refer to the course reference book. [15]

V. Digital Control and Real-Time Digital Signal Processing based System Design

Computers have become ubiquitous in industrial and educational use. Digital Control and Digital Signal Processing are two very important computer-based disciplines. The knowledge of both of these disciplines is essential for our students. Digital Control is used in many industrial processes because it is economical, implementable and re-configurable on demand. The main applications of DSP are audio signal processing, audio compression, digital image processing, video compression, speech processing, speech recognition, digital communications, digital synthesizers, radar, sonar, financial signal processing, seismology and biomedicine.

The Engineering Technology / Electrical and Computer Engineering Technology (ECET) program at our University currently has a two-course sequence of ECET 38400 and ECET 39200 to teach Digital Control and the Digital Signal Processing respectively. Both the courses require background in discrete-time signals and systems, conversion from analog to digital domain, and Z-Transform. This background is provided in ECET 38400. As such, this course is prerequisite for ECET 39200. In ECET 38400, after the introduction of the basics, the course focuses on the stability of systems and feedback leading to the design of digital controllers. MATLAB / SIMULINK engine is utilized extensively in the course textbook [16].

In the second course, Digital Signal Processing ECET 39200: After reviewing the basics, the course focuses on the Discrete Fourier Transform, Fast Fourier Transform, design of digital filters, and its implementation using a digital signal processor TI C6748 on OMAPL138 – LCDK Development board. Students learn real-time voice processing through C programming. Other topics include Multi-Sampling, Image Processing and introduction to Wavelets.

The ECET program faculty feels that even though the current two-course sequence provides the students with a good understanding of the methodology and principles, it falls short of making them productive as designers of Digital Control and DSP systems. One way to address this is by offering graduate-level courses in both areas.

VI. Pedagogy of the Courses

The origin of OBE as an established pedagogical methodology was set forth by the signing of the 1989 Washington Accord, and the faculty members in our department subscribe to OBE. The basic philosophy of OBE is in an instructor's design and delivery of a course. In order to be fully outcome-based, the instructor has to be cognizant of the fact that a course must be organized, such that (1) outcomes are fixed and (2) time and other resources have to be accordingly arranged. The department's courses have culminated in a Team Final Project which is assessed based upon its course outcomes, comprehensiveness and originality. Students are required to master (1) the soft skills of comprehensive report writing on a weekly basis, (2) Technical Project Report writing and (3) an oral presentation based upon the Team's Final Project. All classes meet in the studio/lab environment. This interactive learning environment not only provides students with content, but also with context, and thus prepares them for a lifelong learning process and career path.

VII. Conclusion

The curriculum changes discussed here that are in practice currently by the Electrical and Computer Engineering Technology Department at Purdue University Northwest have integrated the areas of disciplines that make our students industry-ready: PLD/FPGA centric system design, Microcontroller based Embedded System Design, PC based / or System on the Chip (SoC) based systems like Beagle Bone Black based network-oriented distributed System Design and IoT and cloud centered programming and DSP based Real-time Processing based System Design. An important point to note is that throughout the curriculum, each of the four disciplines has at least two courses, one taking the students from an introductory level to a level of mastery, then the other preparing them to transition smoothly into industry positions. All these courses in their present state rely heavily on software, as software-hardware integration is a primary component of all these courses; as such, this paper serves as a pointer for fellow academicians to incorporate its aspects into their curricula.

Bibliography

[1] O. Farook, C. R. Sekhar, J. P. Agrawal, E. Bouktache, and A. Ahmed, A., Purdue University Calumet, Hammond, Indiana, M. Zainulabeddin, Electronics Corporation of India Limited, ECIL Post, Hyderabad, India. "Electrical and Computer Engineering Technology Curriculum from the System Design's Perspective", Proceedings of the 2004 American Society for Engineering Education Annual Conference, June 20 - 23, 2004. Salt Lake City, Utah.

[2] W. Raslan, "Logic Circuit Families", Electronics ECE 39 Sinai University.

[3] S. Lakeou, T. Dinh, and A. Negede, "PRACTICAL DESIGN PROJECTS UTILIZING COMPLEX PROGRAMMABLE LOGIC DEVICES (CPLD)", University of the District of Columbia, AC 2007-2485.

[4] Microchip corporation: http://www.microchip.com

[5] Design with PIC Microcontrollers, John B. Peatman, ISBN: 0-13-759259-0, Publisher: Prentice Hall, Copyright: 1998

[6] http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

[7] J. Purum, Beginning C for Arduino. Ecosoft, Inc. Cincinnati, Ohio. ISBN 13: 978-1-4842-0941.

[8] https://learn.adafruit.com/adafruit-arduino-selection-guide/arduino-comparison-chart

[9] https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/all; https://cdn.sparkfun.com/datasheets/Kits/SFE03-0012-SIK.Guide-300dpi-01.pdf

[10] W. Yu, O. Farook, J. P. Agrawal, and A. Ahmed, A. (2019, June), "Teaching Microcontrollers with Emphasis on Control Applications in the Undergraduate Engineering Technology Program", 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/27895

[11] http://beagleboard.org/hardware/design

[12] https://www.raspberrypi.org

[13] https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/installation-on-ubuntu

[14] O. Farook, J. P. Agrawal, A. Ahmed, A. Kulatunga, N. K. Koyi, H. A. Alibrahim, M. Almenaies, (2016, June), “Embedded System Design Based on Beaglebone Black with Embedded Linux”, 2016 ASEE Annual Conference & Exposition, New Orleans, Louisiana. 10.18260/p.26927

[15] D. Molloy, Exploring Beagle Bone Tools and Techniques, John Wiley & Sons Publication 2015.

[16] J. P Agrawal, FIRST COURSE IN DIGITAL CONTROL: USING MATLAB/SIMULINK, Create Space Independent Publishing Platform (August 8, 2016) ISBN-13: 978-1533086334