

Visual Embedded System Programming Has Arrived!

Alex Doboli, Simona Doboli and Edward H. Currie
adoboli@ece.sunysb.edu, Simona.Doboli@hofstra.edu, ehcurrie@resonance-pub.com

Abstract - Many ECE graduates have only limited experience with design and integration of mixed-signal, embedded systems. This is in part due to the multitude of topics that must be taught to students during a very limited period of time. Fortunately, recent advances in reconfigurable hardware/software platforms and Visual Programming tools have the potential to dramatically address this deficiency, and enable a paradigm shift in ECE undergraduate education towards complex systems and networks of systems. This paper presents new educational material based on Visual Programming and reconfigurable mixed-signal systems on a chip to teach mixed-domain co-design and integration, including analog, digital, and software. The paper presents a new introductory course for freshmen ECE students, which is currently being developed at Stony Brook University, and a senior design project that was successfully completed at Hofstra University.

Index Terms – educational material, embedded systems, mixed-signal embedded architectures, Visual Programming

INTRODUCTION

Mixed-signal, embedded systems are projected to represent 25-50% of the semiconductor market applications by 2011. Current Electrical and Computer Engineering (ECE) curricula include an introduction to topics such as control system theory/design, analog and digital circuits, embedded systems, and microprocessors. However, industry feedback suggests that the majority of graduate engineers have, at best, only limited experience with system design and integration of mixed-signal, embedded system design. This deficiency is in part due to the multitude of theoretical topics and practical skills that must be taught to students during a very limited period of time. Fortunately, recent technological advances have resulted in reconfigurable hardware/software platforms and Visual Programming tools that have the potential to dramatically address this serious deficiency. Moreover, this development makes possible a paradigm shift in ECE undergraduate education from the current focus on circuit design towards design of systems and networks of system [5, 7, 8], which are at the core of modern applications and innovative technological trends [6, 9].

State-of-the-art system on a chip (SOC) technology provides, in a single chip, microcontroller, permanent and

RAM memory, and reconfigurable analog and digital blocks [1], e.g., analog to digital converters (ADCs), digital to analog converters (DACs), timers, counters, filters, instrumentation amplifiers, programmable gain amplifiers, generic switched-capacitor blocks, digital buffers, E2PROM, sleep timers, LCD/LED displays, pulse width modulators, random sequence generators and support for as I2C, IRDATx/Rx, SPIM and UART protocols. As a result, it is now possible to build complex, mixed-signal, embedded application utilizing a single reconfigurable SOC [1, 9]. Low manufacturing costs, low power requirements, rapid prototyping, code reusability, and real-time reconfigurability are additional benefits.

In addition to a comprehensive development environment for assembly and C language app development, a Visual Embedded Programming Tool, allows students to create, and simulate arbitrarily complex, mixed-signal, embedded system applications without writing a single line of either assembly or C language source code [10].

This paper presents new and innovative educational material based on the Visual Programming tools and reconfigurable mixed-signal SOCs developed by Cypress Semiconductor Corporation [1]. The emphasis of the material is on mixed-domain system co-design and integration, including analog, digital, and software sub-systems as well as networks of such sub-systems. In addition, the material stresses the importance of performance constraints (e.g., cost, speed, reliability, power and energy consumption, etc.), design trade-offs, and scalability and easy-access through networking and web based applications. The paper presents a new introductory course for freshmen ECE students, which is currently being developed at Stony Brook University (SUNY), Department of ECE, and a senior design project that was successfully completed at Hofstra University, Department of Computer Science. This material is, in addition to our previously developed undergraduate and graduate mixed-signal, embedded system design course supported by a textbook and companion lab manual [9, 11]. This work has provided the basis for several research papers and application notes co-authored by students. Student reaction has been excellent.

This paper has the following structure. Section 2 introduces the main elements of the visual programming notation and the related reconfigurable architecture. Section 3 presents a new introductory freshman course on mixed-signal embedded systems, and a senior design project is discussed in Section 4. Finally, conclusions are offered.

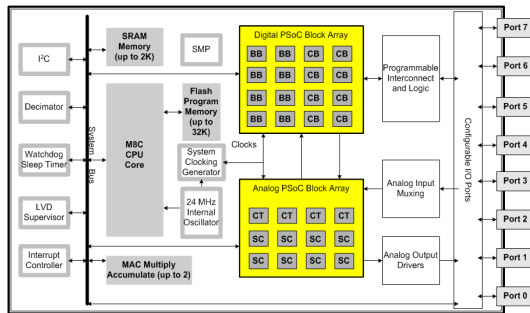


FIGURE 1: PSoC MIXED-SIGNAL EMBEDDED CONTROLLER

VISUAL PROGRAMMING IN PSoC EXPRESS

Cypress recently introduced a quantum leap in traditional microcontroller design by incorporating not only all of the features of the microcontroller but a broad suite of reconfigurable hardware modules, e.g., in addition to ADC and DAC modules, there are PWMs, UARTs, filters, counters, timers, MUXs, random sequence generators, LED/LCD/7 segment display drivers, sleep timer and modules that support I2C master/slave protocols (see Figure 1) [1]. These modules can be combined, i.e., “wired”, by the designer at design time and, in addition, be “re-wired” (reconfigured) in real time under programmatic control using a concept called reconfigurability. The resulting Cypress technology is called PSoC, or Programmable System on Chip, and is available in a variety of packaging formats, which are an upwardly compatible architecture. This allows a designer to “migrate” designs as the requirements specifications become more complex, e.g., as a result of market demand. By combining so much capability in a single package the cost, power requirements, PCB real estate requirements, physical volume, etc., can be minimized. In addition, unwanted coupling between devices can be minimized.

Even the most cursory of reviews of the history of embedded system core design shows that while the technology for development of more and more complex cores has been possible primarily as a result of the rapid evolution of increasingly more sophisticated development tools, such has not been the case for the development of embedded systems based on such cores. Assembly language and C language-based tools have remained the primary vehicles for such design. As the requirements for embedded systems have become more and more complex the concomitant design challenges have increased correspondingly. However, a typical development environment still consists of an assembler, linker, debugger and often a C compiler.

While much attention has focused on the myriad benefits attributed to code reuse, the available development tools are not particularly well-suited for the creation and sustaining of reusable software modules. Furthermore limited memory and performance requirements have tended to encourage the use of low level tools in order to optimize

the use of the available resources. It should also be noted that many designers have insisted on the ability to optimize any code used in their designs. As core processor speeds have increased and memory resources have expanded, it has become feasible to evolve embedded system development tools to not only ease the designer’s development burden but also to make code reuse practical while meeting the ever increasing market demands for lower cost, faster performance, minimal component count, shorter time to market, etc. Once code reuse becomes feasible it is possible to significantly increase the level of abstraction employed by development tools, and thereby reduce the designer’s involvement in the lower level implementation details.

An excellent example of such an approach is illustrated by a development environment known as PSoC Express. This tool is based on an integral code generator that utilizes code fragment libraries that contain pre-defined and tested software modules to perform a wide variety of input/output, signal conversion/processing and networking functions based on protocols such as I2C. In addition, PSoC supports an important concept known as reconfigurability which refers to the embedded system’s ability to complete re-configure itself in real time to obtain the maximum use of the available resources.

PSoC Express operates above Cypress’ more traditional development environment, PSoC Designer. Once a design has been defined in PSoC Express the resulting source code is then passed to PSoC Designer which transparently “builds” an executable.

The input device library includes support for acceleration, airflow, CapSense, compass heading, current, digital input, distance, expanders, humidity, light sensors, pressure, remote devices, resistance, speed, tactile, temperature, timing, voltage and a variety of other input devices. The output device library support includes banked I/O, digital output, display, expanders, fans, high brightness LEDs, PWMs, I2C slave/master configurations, wireless USB, CPU reset, stochastic signal density modulators, timing, voltage output, etc.

The input and output functionality is defined and controlled by logic embodied in table lookups, state machines, priority encoders, status encoders, loop delays, etc. The designer selects the input and output devices and then sets up, the state machine, table look up and/or other logic required to provide the necessary functionality. A simulation mode is also provided which allows the designer to test the system’s logic. Once the design is completed and the logic tested, the designer can then invoke PSoC Express’s “Build” command and create the associated firmware. It is at this step in the development process that the designer selects the target hardware and makes the pin assignments.

As an illustrative example, consider the design of a PWM (Pulse Width Modulator) motor controller which utilizes a potentiometer as an input device and a PWM as the output device (see Figure 2). The designer selects the input and output devices from the available libraries and then

selects a priority encoder to implement the necessary I/O transfer function. This encoder employs conditionals in the form of If-Then-Else statements to implement the necessary logic. Next the designer selects an I2C slave interface to monitor the design from a PC.

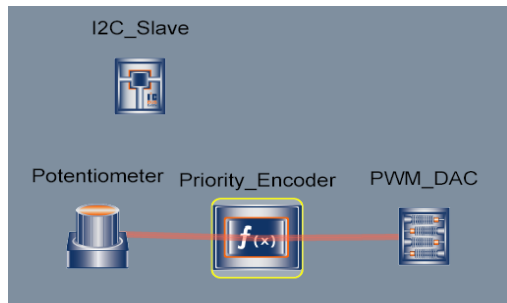


FIGURE 2: PSoC EXPRESS DESCRIPTION

The design can then be tested by using PSoC Express's simulation mode. The designer can vary the input parameters and observe the corresponding output parameters to confirm correct operation.

Once it has been confirmed that the design is correct, the designer invokes the build option, selects the target hardware, and assigns the I/O pins (which can be selected automatically or by the designer). When the build has been successfully completed a BOM, data sheet and schematic are generated. The designer then uses PSoC Express's integral programmer to download the firmware to the target system.

PSoC Express also provides a "Monitor" function which allows the designer to "tune" the design for optimum performance by providing access to the working design. The designer can also invoke Cypress' lower level development environment, PSoC Designer, and examine the design's source code. This capability allows the designer to optimize the design if desirable and add support for functionality or devices not currently supported by PSoC express. While changes to the code produced by PSoC Express by a tool such as PSoC Designer, can not be further processed by PSoC Express, it does address the requirement of designers who insist on access to all of a design's source code and the ability to modify it.

The designer can also add components to PSoC's standard libraries. Each such driver is based on three elements: source code, metadata and code fragments and the user interface together with the associated documentation. There are three basic types of drivers supported by PSoC Express: input, output and interface (I2C, SPI, RS-232 and USB). Once a new component has been designed it can then be integrated into PSoC Express on the same footing as the standard library components. Fortunately, the standard libraries support a wide variety of commonly used devices so that often it will not be necessary to create components in order to implement a design.

PSoC Express also supports an important new technology known as CapSense which utilizes capacitive sensing to replace many of the traditional input devices such as switches, sliders, etc. Capacitive sensing replaces not only

mechanical buttons and sliders but also membrane technologies in cell phones, iPods, cars, PCs, etc. This technology is resistant to environmental factors such as water, temperature and humidity.

Visual Programming is based on compositional software development rather than on procedural algorithms like in traditional design. Programming includes identifying the building blocks of an application and then interconnecting the blocks to produce the data flow of the application. Hence, emphasis is set on designing the structure of an implementation rather than representing that structure in a programming language. Algorithmic procedures are mostly embedded in the descriptions of the building blocks and can be reused for similar applications. Also, there is less emphasis on data structures as variables and data types are transparent to the programmer and are automatically selected by the programming environment. This is important for embedded systems which might involve a large variety of electrical signals and data. Programs are illustrative, easy to understand and modify.

The following two sections describe educational material that has been developed based on PSoC mixed-signal embedded microcontroller and PSoC Express design environment.

USING VISUAL PROGRAMMING FOR INTRODUCING FRESHMEN TO ECE TOPICS

This section describes an introductory course that we are currently developing for introducing freshman students to the main topics in Electrical and Computer Engineering (ECE). The course relies heavily on the visual programming tool PSoC Express and the reconfigurable mixed-signal controller PSoC. The goal of the course, entitled "Introduction to Electrical and Computer Engineering", is to present the basic theoretical concepts in ECE, and to illustrate them through hands-on laboratory experiments.

Introductory courses with similar goals are offered by the large majority of ECE programs. Traditionally, these courses focus primarily on the theory of basic analog and digital circuits, e.g., simple amplifier circuits, basic filters, and small combinational digital circuits [3, 4]. Introductory signal processing algorithms are sometimes covered; however, teaching simple analog and digital circuits to students remains the primary focus of the traditional courses.

Visual Programming for mixed-signal, embedded architectures creates the opportunity to introduce students to a much broader set of ECE concepts, so that the course emphasis is shifted from mainly circuit related topics to system design and networks of embedded systems. The use of Visual Programming is critical in this paradigm shift because otherwise it would be very hard to present sufficient details of such complex topics to freshman students.

The objectives of the new introductory course is to teach the following fundamental concepts to students: electrical signals, data, embedded processing, performance constraints, design trade-offs, sensors, basic circuits in analog front ends, digital circuit fundamentals, embedded

software, networks of systems, and web applications. The course also includes discussion of some of the advanced topics that are currently being investigated by the ECE research community, such as nanodesign, biocomputing, robotics, artificial intelligence, medical applications, energy harvesting, etc. The laboratory activities include hands-on experiments related to the topics presented.

Concepts are introduced through simple, illustrative examples, and the defining characteristics for each concept are summarized and discussed. Intuitive design, analysis, and testing procedures are also presented. Using Visual Programming allows a more intuitive presentation of the essence of the main concepts without having to utilize complicated specification notations, such as programming languages, or complex circuit schematics, which can be difficult for freshmen students to understand.

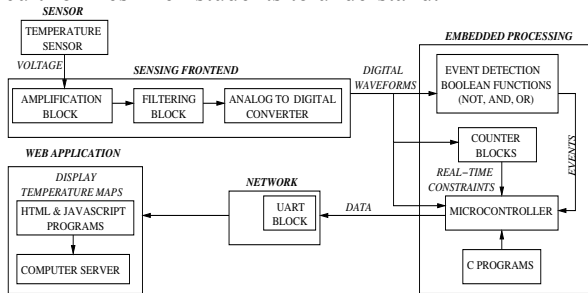


FIGURE 3: INTEGRATION OF THE COVERED TOPICS

The course material uses an integrated application for putting each of the fundamental topics into context, and to explain the correlations with other important ECE or cross-disciplinary areas. The goal of this application is to produce a real-time temperature map of a geographical zone which can be displayed on a computer. Figure 3 depicts the basic building blocks of the application, and the fundamental concepts that are presented using the blocks as a vehicle.

The 14 week-based course structure is as follows:

- **Introduction** (1 week): Description of various ECE applications in telecommunication, manufacturing control, automotive, consumer markets, entertainment, etc., including their defining characteristics, challenges, and markets.
- **System description** (3 weeks): Presentation of electrical signals (currents, voltages, conservation laws), mapping of electrical signals to digital data, conversion of analog signals to digital data and digital data to analog signals, data acquisition and control system building blocks (e.g., front ends, embedded processing, networking), performance specification (speed, cost, precision, reliability, bandwidth, power and energy consumption, etc.), design trade-offs (such as cost – speed – power/energy consumption), and design and implementation alternatives.
- **Sensing front ends and actuation** (3 weeks): Discussion of different sensor types and analog front ends, including basic sensing principles for popular sensors such as temperature and capacitive sensors,

Global positioning systems (GPS), RFID, imagers, and the related analog building blocks for signal conditioning, filtering, and analog to digital conversion. An intuitive design is presented for each building block as well as its defining parameters, like gain, stability, bandwidth, precision, etc.

- **Embedded processing** (4 weeks): This module focuses on data processing in hardware and software. A simple real-time algorithm is discussed which explains the importance of digital hardware and software. Counters, logic gates and other digital circuits, are used to detect events coming from the sensing front ends and to monitor the timing constraints. This algorithm illustrates some of the main programming concepts, such as variables, instructions, programming languages, etc.
- **Networks of systems and web programming** (2 weeks): This module will intuitively define the main networking concepts, and introduce students to simple web applications, such as simple HTML and JavaScript programs, SQL, and security issues.
- **Future trends** (1 week): The course concludes with a discussion of some of the modern trends in ECE, including nanodesign, biocomputing, artificial intelligence, robotics, etc.

For each of the topics, the course enumerates the more advanced, related concepts, and how they are further explored by other courses in the ECE curricula.

The laboratory activities include the following experiments performed on the PSoC mixed-signal embedded architecture:

- **System description** (2 weeks): Specification of the real-time temperature mapping system, including functionality, and performance constraints in PSoC Express.
- **Sensing front ends** (5 weeks): Interfacing temperature sensors to PSoC; measuring the sensor output voltage; plotting the characteristics of the sensor; connecting the sensor to an amplification stage based on PSoC's reconfigurable analog blocks; changing the amplification of the block; connecting an ADC to the front-end; and relating the digital data to the observed ADC waveforms and to the sensor voltage output, intuitively.
- **Embedded processing** (5 weeks): Building a counter module using the reconfigurable digital PSoC blocks; developing a simple C program for PSoC's microcontroller; and connecting the counter to the microcontroller.
- **Networks of systems and web programming** (2 weeks): Transmitting data to, and from, a central server to embedded boards. Writing a simple web program to display the temperature map of a zone.

The course grading is based on two in-class exams (approximately 70% of the final grade) and the laboratory activity (approximately 30%). We are also considering the

possibility of having a course project to offer more hands-on training to students.

The next sections describes a senior design project that was performed at Hofstra University by students E. Hjelm and R. Cheng [10].

4. SENIOR DESIGN USING VISUAL PROGRAMMING

The project’s aim was to design an alert system which reacts to human or animals left in parked cars under very high temperatures. To sense a potentially dangerous state, the system used two sensors: a temperature and a motion sensor. If the temperature was higher than 38°C, and the motion sensor’s input was higher than 80%, the system would switch to alarm state. In this state, the system would react in two ways, first it would signal the car’s dangerous situation by turning on the car alarm and blinking a strobe light, secondly it would reduce the inside temperature by starting the car’s engine and air conditioner, remotely.

The system’s architecture is presented in Figure 4:

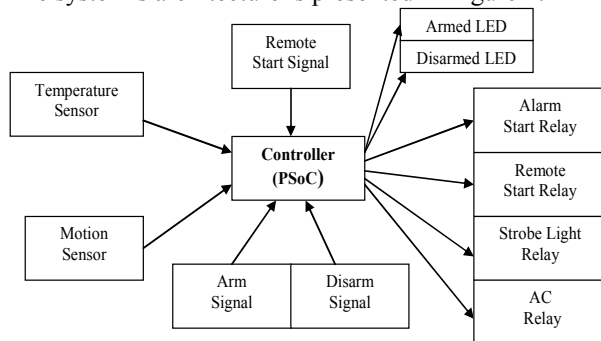


FIGURE 4: THE ARCHITECTURE OF THE CAR ALARM SYSTEM

The controller of the car alarm system was implemented with a PSoC Evaluation Kit CY3210-Eval. The inputs were the two sensors and the remote unit: the remote start, the arm and disarm signals. The outputs were two LEDs to indicate the state of the system: Armed or Disarmed and outputs to turn on the alarm, remote start, strobe light and air conditioner.

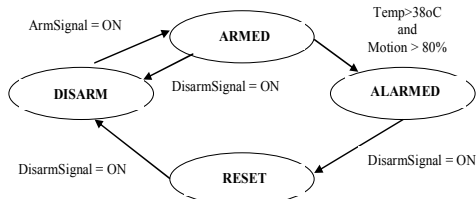


FIGURE 5: FSM DIAGRAM FOR THE CAR ALARM SYSTEM

The system functions as follows: if the car is parked, and the remote unit arm signal is pressed, the system goes into the Armed state (the ArmedLed turns on). In this state, it monitors the motion and temperature sensors, and if the temperature is too high and sufficient motion is detected, the system switches into the Alarm state. In this state, the visual and audio alarm signals, AC and the car’s engine are turned on. If the disarm signal, on the remote unit, is pressed while

the system is in Alarm state, the system goes into a Reset state in which the both Armed and Disarmed LEDs blink so that the owner is aware that the alarm went off. To return to the Disarmed state, the disarm signal must be pressed a second time. The FSM of the car alarm system is shown in Figure 5.

The pseudo-code of the control algorithm implementing the FSM in Figure 5, is next:

```

while (true) {
    ArmLED = OFF;
    DisarmedLED = OFF;
    STATE = DISARM;

    if (DisarmSignal >= 3V) {
        DisarmedLED = ON;
        STATE = DISARM;
        if (RemoteStartSignal > 3V)
            CarStart = ON;
    }
    else
    if (ArmSignal >= 3V) {
        //alarm is armed
        ArmedLED = ON;
        STATE = Armed;

        if ((TempSensor >= 38°C) &&
            (MotionSensor > 80%))
            STATE = ALARM;

        if (STATE == ALARM) {
            StrobeLight = ON;
            AlarmStartRelay = ON;
            RemoteStartRelay = ON;
            ACStartRelay = ON;

            if (DisarmSignal > 3V) {
                // alarm is disarmed
                STATE = RESET;
                DisarmedLED = blink;
                ArmedLED = blink;
            }
        }
    }
    if (STATE == RESET)
        if (DisarmSignal > 3V)
            STATE = DISARM;
    } // if
} // end while loop

```

The project’s design was implemented using PSoC Express [10]. The FSM in Figure 5 was entered in PSoC Express as shown in Figure 6. The implemented state machine has five states. The extra state is actually a double Reset state: ResetA and ResetB. The ResetA was introduced just to add a delay in the system. The delay is needed because the DisarmedLED flickers after it has been pressed once. This occurs because the disarm pulse from the alarm’s microcontroller is that of an extended pulse. And since it is extended, there is a need for a delay. In order to bypass this delay, ResetA state was configured so that the microcontroller would see the trailing edge of the pulse and use that as the basis for transitioning to another state, instead of checking when the signal has reached zero volts.

In addition to the five states, the remote starter output was integrated into the system to allow for the remote starter to work without the PSoC microcontroller. This was done by creating more states that would come from the Disarm

state. Since the remote starter required two pushes of the remote start button on the key fob to start the engine, two states were implemented to get to the RemoteStartBypass state so as to catch the falling edge of the pulse of the second pulse. And to turn off the remote starter, the remote starter controller needs two additional pushes of the buttons on the key fob. Thus another state was created to catch the negative edge of the second pulse to signal the turning off of the remote starter. Thus with these new additions, the system now functions correctly according to its goal.

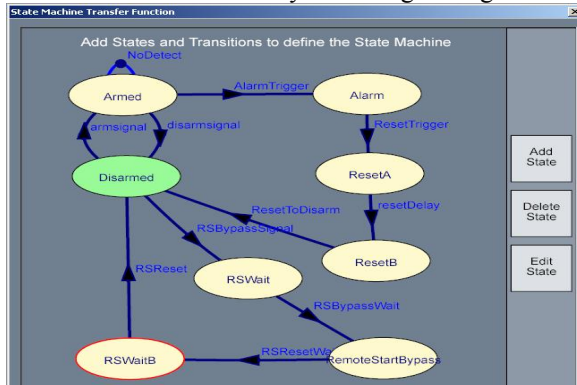


FIGURE 6: FSM IMPLEMENTATION ON PSoC

PSoC Express was used to verify and simulate each state transition. The whole system was implemented and tested. Figure 7 shows the system's simulation when it transitions from Armed state to Alarm state.

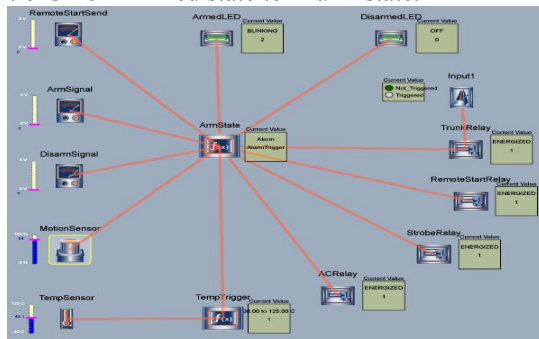


FIGURE 7: PSoC Express SIMULATION OF THE ARMED TO ALARM STATE TRANSITION: ArmSignal > 3V, State = Armed AND TempSensor > 38°C AND MotionSensor > 80%, THEN State = Alarm, ArmLED IS BLINKING

Discussion. PSoC Express and PSoC allowed the students to implement the relatively complex design of the car alarm system in a relatively short period of time. The functionality of the system was described graphically as a finite state machine. The input and output signals were also easily specified based on the type (analog or digital) and the range of values (for analog) and mapped to the chip pins. Once the system was fully specified, PSoC Express automatically generated the executable code. The system could then be fully simulated and tested in software, which allowed errors to be fixed without affecting the hardware components (i.e. the PSoC chip and any other connecting circuits). Once the system was verified, the executable was downloaded to the

evaluation board. An advantage of the PSoC evaluation board is the convenience of using it as a stand-alone system.

CONCLUSIONS

Recent advances in reconfigurable hardware/software platforms and Visual Programming tools have enabled a paradigm shift in ECE undergraduate education towards mixed-domain systems and networks of systems. This paper presents new educational and innovative material based on the Visual Programming tool, PSoC Express, and the reconfigurable platform, PSoC, to teach mixed-domain system co-design and integration, including analog, digital, and software. The material discussed in this paper includes a new introductory course for freshmen students, which is currently being developed at Stony Brook University, and a senior design project that was successfully completed at Hofstra University. In our experience, PSoC Express and PSoC allowed the students to implement, debug, and experiment relatively complex designs in a relatively short period of time. Also, PSoC-based designs are convenient to use either as stand-alone systems, or integrated with other optical or mechanical systems.

REFERENCES

- [1] "PSoC Mixed Signal Array", Technical Reference Manual, Document No. PSoC TRM 1.21, Cypress Semiconductor Inc., 2005.
- [2] R. Ohba, "Intelligent Sensor Technology", J. Wiley, 1992.
- [3] "Accreditation Policy and procedure Manual", <http://www.abet.org>.
- [4] "IEEE CS/ACM Computing Curricula & Computer Engineering", <http://www.eng.auburn.edu/ece/CCCE/WoodenManReport.pdf>, 2003.
- [5] H. De Man, "System-on-chip Design: Impact on Education and Research", IEEE Design & Test of Computers, July-September 1999.
- [6] D. Estrin, D. Culler, K. Pister, G. Sukhatme, "Connecting the Physical World with Pervasive Networks", IEEE Pervasive Computing, Vol. 1, No. 1, pp. 59-69, January-March, 2002.
- [7] A. Laffely, W. Bursleson, "Using Systems-on-a-Chip as a Vehicle for VLSI Design Education", Proc. ME Confrence, 2002.
- [8] J. Nestor, D. Rich, "Adding Analog and Mixed Signal Concepts to a Digital VLSI Course", Proc. ASEE Annual Conference, 2002.
- [9] A. Doboli, E. Currie, "Introduction to Mixed-Signal Embedded Design", textbook, to appear, 2008.
- [10] E. Hjelm and R. Cheng: Computer Engineering Senior Design Project Report, Hofstra University, May 2006.
- [11] A. Doboli, Design using Programmable Mixed-Signal Systems-on-Chip, <http://www.ece.sunysb.edu/~adoboli/ESE366.html>.

AUTHOR INFORMATION

Alex Doboli, PhD, Associate Professor, State University of New York at Stony Brook, Department of ECE, Stony Brook, NY, 11794-2350, adoboli@ece.sunysb.edu.

Simona Doboli, PhD, Associate Professor, Director of CE Program, Hofstra University, Department of CS, Hempstead, NY, Simona.Doboli@hofstra.edu.

Edward H. Currie, PhD, Resonance Publications Inc.