AC 2007-1475: UPGRADING A MICROCONTROLLER SYSTEMS COURSE WITH THE CYPRESS PSOC

Kevin Bolding, Seattle Pacific Univ

Dave Van Ess, Cypress Semiconductor

Upgrading a Microcontroller Systems Course Using the Cypress PSoC

Abstract

As our society continually embraces technology to greater and greater degrees, the need for engineers with the skills to develop microcontroller based systems is constantly growing. Microcontroller systems design courses are meant to equip students with the understanding, skills, and methods needed to design and develop systems built around a microcontroller core. When designing such a course, the selection of a particular microcontroller is a very important decision. The selection should consider not only what microcontrollers are currently popular, but also the ease of project development using the system, the availability of support to students, the costs of starting up a lab, and the flexibility of the platform to fit into a course with multiple objectives. This paper reports on using Cypress Semiconductor's Programmable System on a Chip (PSoC) as the basis for a microcontroller systems design course. The experience of selecting the PSoC, designing a curriculum around it, designing laboratory exercises and managing the course are described. Furthermore, considerations such as the technical and financial support provided by Cypress and student experiences and outcomes are considered and analyzed. Finally, achievement of students who had taken the PSoC course or an earlier non-PSoC microcontrollers course are compared based on their performance in the follow-on senior design course.

Introduction

Small embedded systems based on inexpensive microcontroller cores have pervaded modern societies to the point that most people in industrialized nations interact with dozens of such systems on a daily basis. Developing small systems based on microcontrollers can be both enjoyable and profitable – both hobbyists and professionals enjoy the endless possibilities and inexpensive solutions available to the clever designer. The practical nature and powerful potential of microcontroller systems can form the basis for a very successful course in microcontroller systems design. Such courses are often found in undergraduate Computer Engineering, Electrical Engineering and Computer Science curricula, usually offered during the junior or senior year of study. A successful microcontroller systems design course will normally include a large amount of hands-on development and experimentation with a physical development system. Ideally, a course would include experimentation with a variety of different types of microcontroller systems. However, because of time constraints in the class and the experiment with systems to a reasonable degree of depth, most courses are centered on a single development system. The choice of development system can significantly impact the success of the course and the applicability of the course material to practical use in internships and post-graduate employment.

In this paper, we examine several factors that can influence the choice of microcontroller development system for a course. We then chronicle the specific case of changing the development system in Seattle Pacific University's Microcontroller Systems Design course from one based on an older microcontroller to one based on the Cypress Programmable System on a

Chip (PSoC) and evaluate these based on the selection criteria. Finally, we provide early results on the differences seen in student performance in the senior design course taken by students after completing the microcontroller systems course.

Microcontroller systems courses

Microcontroller Systems courses (often also called Microprocessor Systems) tend to have very similar formats. Most require programming and logic design as minimum prerequisites. In general, the course will begin by introducing the chosen development system and teaching basic assembly language (or embedded C) programming using the system. Then, once students are up to speed with the system, various components of microcontroller systems such as memory busses and timers are explored one-by-one. In general, for each component explored, students will complete a moderately-large exercise that requires them to use or interface with the component in a laboratory setting. Typically, a course will follow a sequence such as this[1, 2]:

1. Organization of microcontroller systems
2. Introduction to the development system used in the course
3. Programming the development system in assembly language or embedded C
4. External bus (memory) interfacing and timing
5. Exceptions and interrupts
6. Timer system
7. Parallel port interfacing
8. Serial port interfacing
9. Analog-to-digital conversion
10. Brief exploration of alternative microcontroller choices

Because the details of almost every part of the course depend entirely on the type of microcontroller chosen and the development environment provided, it is critical to the success of the course that a careful choice of microcontroller and development system is made for the course.

Guidelines for choosing a microcontroller development system

Upon considering the purposes and structure of a microcontroller system design course, several implications for the choice of microcontroller become apparent. These can be grouped into four categories: Suitability for the class objectives, relevance to real markets, quality of the development environment, and support from the manufacturer.

Suitability for the class structure and objectives

While individual course structures and objectives will vary, most microcontrollers courses are designed to develop skills related to designing and implementing small and medium sized systems that require only a modest amount of computing ability. Key issues for such systems include low cost, low power consumption, the ability of the microcontroller to accomplish many tasks with internal components, and easy integration of the microcontroller with external components. While computing performance is always nice to have, it may easily be traded off for systems that exhibit the other criteria. Thus, we offer the following guidelines in this area:

- Low cost microcontrollers tend to be very suitable for the classroom environment.
- Microcontrollers with reasonably low power consumption are supportive of small systems.
- The microcontroller should include a reasonable variety of internal components that support the course objectives. These should include timers and counters, reasonable amounts of RAM and Flash ROM, and an analog-to-digital converter.
- It should be easy to interface the microcontroller with external components and systems. The microcontroller should include at least one parallel port, one serial port, and a memory interface bus.

Relevance to real markets

Because students spend a significant amount of time learning the how to use the microcontroller and its development system, it is very beneficial to make it easy to leverage this in later class projects, internships, and employment. Choosing a microcontroller that is widely used in industry and is part of a family that includes a wide variety of configurations makes it very easy to transfer the knowledge into practice. Likewise, it seems wise to pick a microcontroller that is neither too new nor too old. The newest families, while having the most advanced specifications, may turn out to be market flops; older families are likely to be behind the times and may have waning market share. The following guidelines have been developed in this area:

- Look for microcontrollers with a reasonably well-developed penetration in the general microcontrollers market. Avoid families that are developed for niche markets.
- Choose a microcontroller family that had an introduction or major update in the range of two to four years ago.
- Look for a microcontroller with a broad family that includes small, low-cost and low-power packages on one end and goes all the way up to large, higher-performance packages on the other end.

Quality of the development environment

Regardless of the quality of the microcontroller chosen, the quality of the student experience will depend much more on the quality of the development environment than on the chip itself. The development environment includes software tools for writing, testing, and debugging code, and for programming and configuring chips. Also hardware components such as evaluation boards, programmers, and interfaces are included in the development environment. The best systems include well-integrated software for performing all of the needed tasks. Ideally, significant quality online help and tutorials should be included. There should be at least one basic evaluation board available; this should include a microcontroller unit, power supply, basic input and output (buttons, LEDs and a simple LCD) and a small assortment of other useful devices such as potentiometers, buzzers or speakers, and keypads. Ideally, a breadboard area will be included to make it easy to interface external components. These lead to the following guidelines:

- Carefully evaluate the development environment before committing to a microcontroller family. Check with alumni and industrial contacts to learn of their evaluation of the quality of the software. Look for well-designed software that provides an integrated development environment.

- Examine the available evaluation boards, including those from third-party manufacturers. Look for a board that will support the various objectives of your course with few external components needed.
- Test the hardware and software systems together. How much training will be necessary before your students are able to compile an example program, load it, and run it on hardware?

Support from the manufacturer

The degree of support from the manufacturer(s) of your development system and hardware components can make the difference between a pleasant and profitable experience and a disastrous exercise in frustration. Most major manufacturers are well aware of the value of university students using their products and will have established university support programs. However, the degree of company commitment to such programs varies considerably. Some are simply avenues to get reduced-cost components to students, while others include full live technical support as well as assistance in developing and setting up laboratories. Companies also vary considerably in what level of support they give to small customers and students. Students who are able to find the help they need in short order will be much more successful. From these observations, we give the following guidelines:

- Look for a company with a well-established university support program. Make inquiries to the staff of the program and gauge the quickness and quality of the responses.
- Most companies that are serious about partnering with universities will provide software at no charge and hardware at or below their cost. Some companies regularly donate equipment to universities outright. Avoid development systems that require significant expenditures on the part of the university.
- Consider the possibility of each student purchasing their own development system. Some companies provide introductory development systems for well less than the cost of a textbook. An obvious advantage is that this allows students to do development work at home – this is more convenient for students and reduces the amount of equipment needed in laboratories. An added advantage is that students who own development systems are more likely to go beyond the course material and experiment with building small microcontroller-based systems on their own.
- Investigate the company you will partner with for the level of technical support they will give to you and your students. Call or email the same support line your students will have and gauge the responsiveness.

The Cypress Programmable System on a Chip (PSoC)

In selecting a microcontroller to form a basis for a course, a key feature to look for is its ability to interface with all aspects of the environment around it. A microcontroller-based system must have an easy and inexpensive interface to sensors, communication interfaces, I/O devices and more. A student learning to design microcontroller-based systems needs to gain knowledge and practical experience in analog and digital design when implementing a complete system. The Cypress Programmable System on a Chip (PSoC) is a highly configurable mixed-signal array

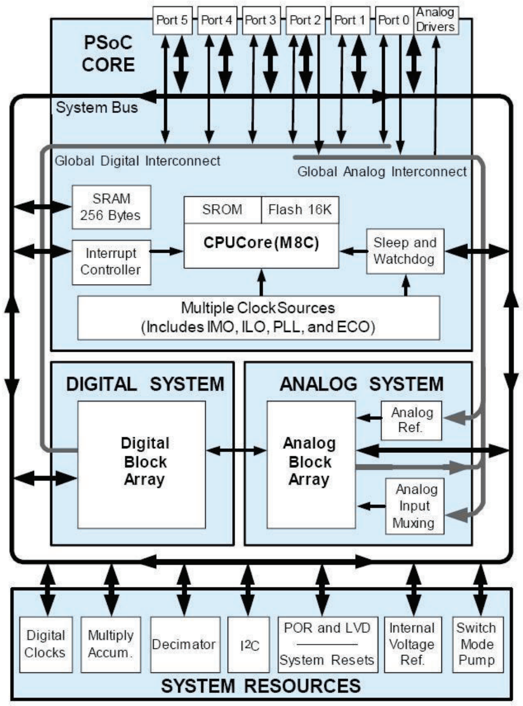


Figure 1: Overview of the Cypress PSoC CY8C27x43 family architecture.

with an on board microcontroller unit (MCU). The PSoC family consists of wide range of devices; the primary differences between models are the size of the mixed signal array, the amount of RAM and ROM memory, and the package size and type. For a microcontroller systems course, the PSoC CY8C27x43 family exhibits a good combination of ease-of-use, resource availability, and applicability to real-world systems. The CY8C27x43 architecture is shown in Figure 1.

The CY827x43 family includes varieties in several different package formats, ranging from 8-pin DIPs to 48-pin SSOPs. A all-around good choice for a microcontrollers course is the CY8C27443-24PXI. It has 24 I/O pins, configured into three eight bit parallel ports, and comes in a 28-pin DIP package. Although DIP ICs are less commonly used in commercial projects, they remain an ideal package for a student building designs with the proto-boards found in the typical student engineering labs. In advanced classes students may decide to select parts with more or fewer resources and using surface mount packages when custom-designed printed circuit boards are available.

The PSoC features four main areas: PSoC Core, Digital System, Analog System, and System Resources. They are discussed in the following sections.

PSoC Core

The PSoC core is designed around a Cypress M8C CPU. The M8C CPU has an eight-bit Harvard microprocessor capable of four MIPS performance at 24MHz. It has an interrupt controller to allow seventeen different interrupt vectors to simplify real time designs. There are 16K bytes of

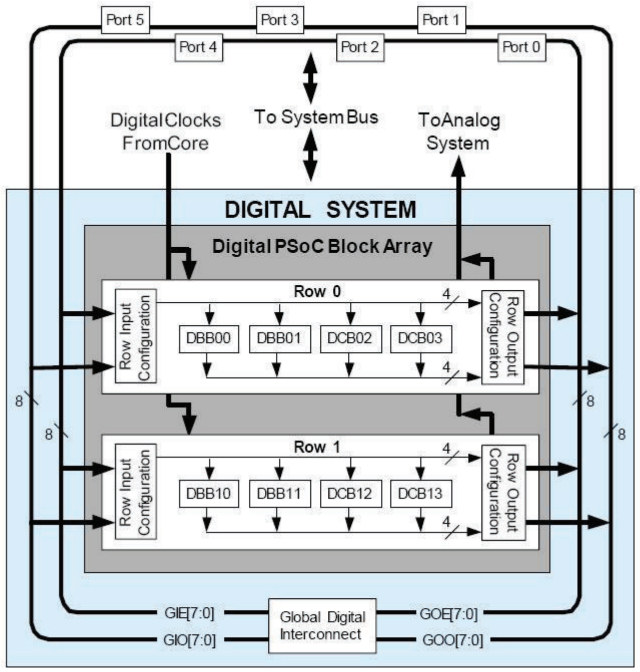


Figure 2: The Digital System of the Cypress CY8C27x43 PSoC family.

program ROM, 256 bytes of data storage RAM, and 241 bytes of system configuration RAM. This is a reasonable amount of memory for small or stand-alone applications. It has an internal 24MHz internal main oscillator, accurate to 2.5% over a standard temperature and voltage range. The main oscillator can also be doubled to 48MHz for use by the digital system. From this main oscillator a CPU clock and three system clocks are be generated. Also included is a sleep and watch dog timer. Each I/O pin is highly configurable and has the capability to generate an interrupt.

Digital System

The Digital System of the CY8C27x43 family is composed of eight digital PSoC blocks - six Digital Basic Blocks (DBBs) and two Digital Communication Blocks (DCBs). The digital blocks are combined into two rows of four each, as shown in Figure 2.

Each digital block is an eight-bit resource that can be used alone or combined to build 16-, 24-, or 32-bit peripherals. The digital blocks and be configured and combined to produce a multitude of user modules, including:

- Pulse-width Modulators (PWMs) (8 to 32 bits)
- PWMs with Dead band (8 to 32 bits)
- Counters (8 to 32 bits)
- Timers (8 to 32 bits)
- Universal Asynchronous Receiver/Transmitters (UARTs)
- Serial Peripheral Interface (SPI) interfaces (both slave and master)
- Cyclic Redundancy Check (CRC) Generators

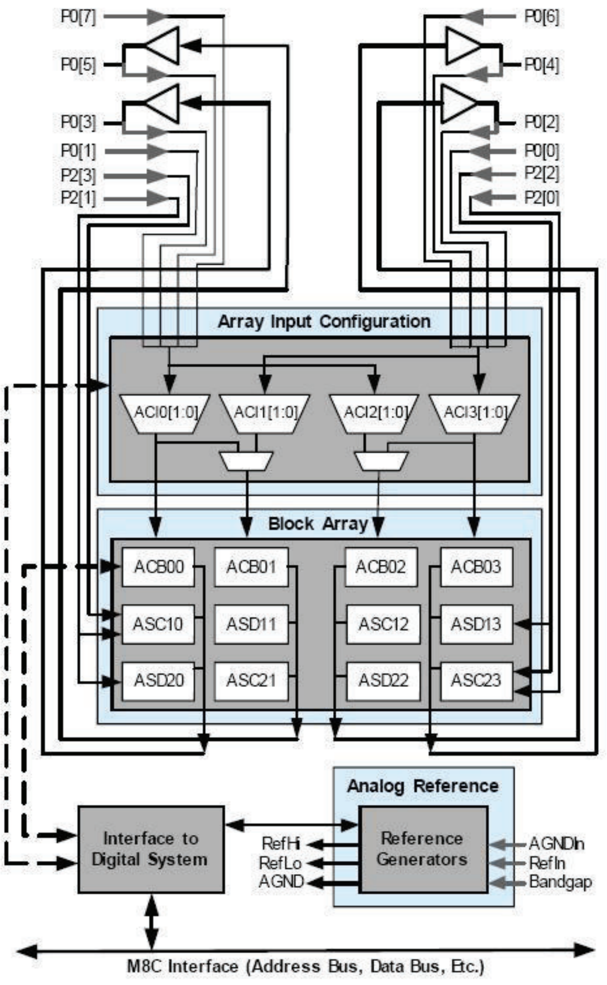


Figure 3: The Analog System of the Cypress CY8C27x43 PSoC family.

- Pseudo Random Number Generators
- Stochastic Signal Density Modulators

Standard user modules (such as those listed above) are easily configured using PSoC Designer (described below). Advanced students may wish to take more control and manually load the digital block configuration registers for other configurations.

Analog System

The CY8C27x43 family's Analog System is composed of twelve configurable blocks, configured into four columns. Each column contains a multiplexer, a continuous time block (ACB), and two switched capacitor blocks (ASCs and ASDs), as shown in Figure 3. Each column also contains an analog output buffer, a comparator to connect the digital section and a programmable data clock.

This topology allows for the creation of unique and complex analog circuits. A combination of these blocks with either each other or with digital blocks can be used to construct the following circuits.

- Analog-to-Digital Converters (6- to 14-bit resolution)
- Filters (low-pass, high-pass, elliptical, and notch)
- Programmable Gain Amplifiers
- Attenuator
- Programmable threshold comparators
- Digital-to-Analog Converters (6 to 9 bits)
- Modulators
- Correlators
- Synchronous Detectors
- Rectifiers

All these circuits allow the student to generate analog stimuli and receive analog input. They can all be implemented with user modules using PSoC Designer. An advantage of this approach over traditional fixed modules is that students may disassemble them to see how the pieces interact. For example, a student can implement only the analog portion of a Delta Sigma ADC to get an intuitive feel of how delta sigma modulation really works.

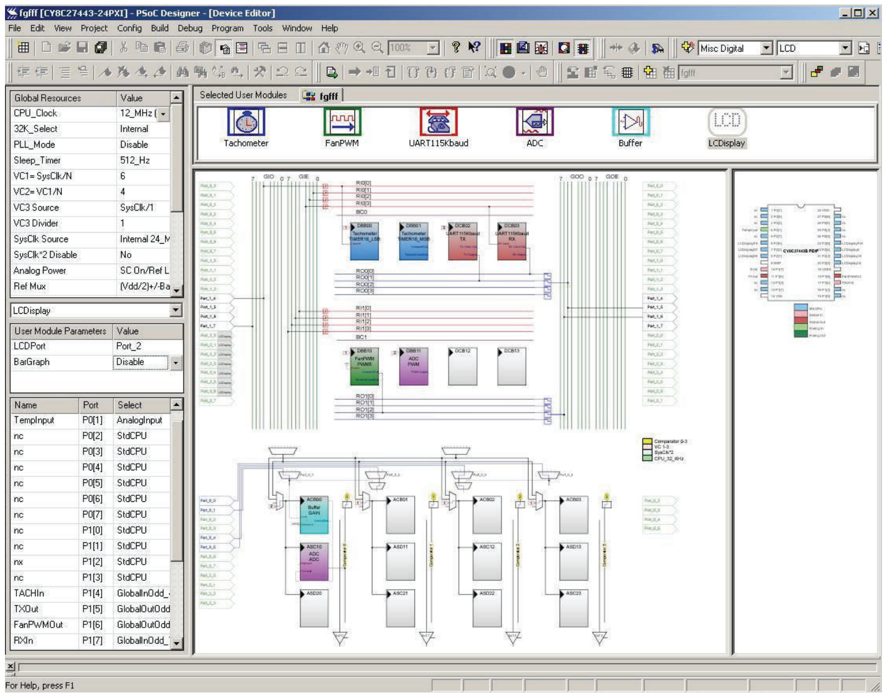
System Resources

System Resources provide additional capabilities that are either useful or required for a complete system.

- Multiply accumulate (MAC) provides fast 8-bit multiplier with 32-bit accumulate, to assist in general math and digital filters. It can be used to teach to fundamentals of basic digital signal processing.
- A digital decimator used as a hardware accelerator for ADC conversion.
- A dedicated I2C engine that provides 100 and 400 kHz serial communication over an industrial standard two wire interface.
- Low Voltage detection circuitry used a system supervisor.
- Internal 1.3V 1% reference.
- An integrated switch mode pump (SMP) boost-converter to generate a normal operating from a supply voltage as small as 1.2V.

Development Environment

PSoC Designer is a fully integrated development environment for all the different families of PSoC parts. It is a Graphical User Interface (GUI) based design tool suite a student can use to configure a design, develop and debug an application. Figure 4 shows the interconnect, or placement, view for a temperature-compensated fan controller.


fgfff [CY8C27443-24PXI] - PSoC Designer - [Device Editor]
File Edit View Project Config Build Debug Program Tools Window Help
Selected User Modules
Tachometer
FanPWM
UART115Kbaud
ADC
Buffer
LCDisplay
For Help, press F1

Global Resources	Value
CPU_Clock	12_MHz (
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	6
VC2= VC1/N	4
VC3 Source	SysClk/1
VC3 Divider	1
SysClk Source	Internal 24_M
SysClk*2 Disable	No
Analog Power	SC On/Ref L
Ref Mux	(Vdd/2)+/-Ba

LCDisplay

User Module Parameters	Value
LCDPort	Port_2
BarGraph	Disable

Name	Port	Select
TempInput	P0[1]	AnalogInput
nc	P0[2]	StdCPU
nc	P0[3]	StdCPU
nc	P0[4]	StdCPU
nc	P0[5]	StdCPU
nc	P0[6]	StdCPU
nc	P0[7]	StdCPU
nc	P1[0]	StdCPU
nc	P1[1]	StdCPU
nx	P1[2]	StdCPU
nc	P1[3]	StdCPU
TACHIn	P1[4]	GlobalInOdd_
TXOut	P1[5]	GlobalOutOdd
FanPWMOut	P1[6]	GlobalOutOdd
RXIn	P1[7]	GlobalInOdd_

Figure 4: PSoC Designer interface. The hardware placement/interconnect view (digital and analog blocks and I/O connections) is shown in this screenshot.

The placement of user modules and their connection to the whole system is graphically displayed using PSoC Designer. Students can program in "C', assembly language or a combination of the two. The debugger suite includes event trigger and multiple dynamic break points. PSoC Designer is free and can be downloaded from http://www.cypress.com/psocdesigner.

The most popular board for PSoC development is the CY3210-PsoCEval1 board. This board includes an LCD module, potentiometer, RS232 drivers, and plenty of bread boarding space. It includes a MiniProg programming unit that connects to the student's computer via an USB cable. Many applications can be developed with only the hardware included with the PSoCEval1 board. More complicated programs may require the use of an in circuit emulator (ICE). The CY3115-DK is a full-featured in-circuit hardware emulator.

Cypress, through its University Alliance, has a generous policy of supporting universities and students through donations and price reductions. Qualifying universities are able to outfit entire laboratories with CY3115-DK in-circuit emulators at no cost. Students are able to acquire complete PSoCEval1 evaluation systems (including an embedded C compiler) for less than half the price of many textbooks. More information can be found at the Cypress University Alliance web page: http://www.cypress.com/cua.

Structuring a course around the Cypress PSoC

It is relatively easy to adapt a typical microcontroller system design course to use the Cypress PSoC. The core functions of the PSoC are similar to many other small microcontrollers and the instruction set is straightforward. For the most part, the course structure outlined above can be applied without difficulty. As with most microcontrollers, there are some aspects of the PSoC that result in unique challenges when using it as the basis for a course. The most significant distinctive features of the PSoC that require tinkering with the course structure are the configurability of hardware blocks within the chip and the absence of a traditional memory bus interface. One other difficultly is the lack of a PSoC-based microcontroller system design textbook. In this section, these issues are discussed and solutions are presented.

The PSoC's configurability is a definite asset for the class, at least when viewed from the medium- or long-term perspective. The ability to mix-and-match a large variety of internal components gives students exposure to and practice with a wide variety of important digital and analog systems. However, in the short-term, there is a problem. Most microcontroller design classes begin with an introduction to writing software for the system. Unfortunately, the very first step in the PSoC design process is to specify the hardware configuration. Although the development system makes this relatively simple, it can be confusing for students who haven't yet written their first line of microcontroller code. The solution to this is straightforward – the instructor develops a PSoC configuration with basic hardware that meets the needs of the first two or three programming exercises. This project is distributed to the class so that students can simply open up the hardware configuration and start writing code. A great starting configuration might include a simple analog-to-digital converter and the Cypress-provided 2x16 LCD driver. With these internal components and the CY3210-PsoCEval1 Evaluation System several interesting, yet easy, exercises are available. For example, novice students can quickly write code to sample the voltage from the PSoCEval1's built-in potentiometers and display this value on the LCD. With the bar graph functions provided by the LCD driver, it is trivial to graphically represent this value on the LCD. Once students have some familiarity with programming the system, the configuration methods can be introduced in a controlled manner. For instance, earlier exercises might describe the configuration options to use, but require the students to enter these. Later, exercises might specify the components and connections to use, but not how to configure them. Finally, as the students become more capable, the hardware design configuration options may be left completely to the students.

The PSoC, like many inexpensive microcontrollers, does not provide an address/data bus for memory expansion. In practice, designers of projects that require additional memory of one type or another will choose parts with two-wire interfaces such as I^2C or SPI. However, most microcontroller systems courses include understanding memory bus timing and address decoding as course objectives. The case may be made that this portion of the curriculum may be moved to a digital systems course, as most low-cost microcontrollers do not include memory busses anymore. However, if it is considered essential that the microcontroller systems course include memory interfacing and address decoding, there are a few options when using the PSoC. One solution is for students to build their own memory bus using many GPIO port pins. However, this solution not only occupies a tremendous amount of hardware space, but requires non-trivial software to implement a bi-directional bus. Moreover, the objective of discussing timing diagrams that show how a single-cycle memory transaction operates gets lost when a single memory read requires nearly a dozen instructions. A more reasonable option is to study the memory interface of an alternative microcontroller or microprocessor. The downsides of this are obvious: The course focus has change to a completely different microcontroller for a week or so in the middle of the course and students will not be able to do hands-on experiments using a memory bus. The temporary switch to study a different system's memory bus is not too difficult to achieve because the bus hardware functionality is mostly isolated from the instruction set – class time can be spent studying the memory bus and its timing without having to delve into the alternative system's instruction set. Also, although students won't have real physical experimentation with a memory bus, such laboratory exercises can be replaced with exercises using memories connected using a two-wire bus instead. While temporarily using an alternative system may not be ideal, it does also have the advantage of exposing students to a different microcontroller/microprocessor system and broadening their understanding of the differences between types of microcontrollers.

The final area of concern is the availability of a quality PSoC-based microcontroller systems textbook. In the area of microcontroller systems design, it is typical to specify a textbook that addresses the course objectives within the context of the selected microcontroller and development system. Currently, there is only one published text that describes the PSoC – *Designer's Guide to the Cypress PSoC*[3] by Robert Ashby. This text is structured as a moderately advanced resource guide for a designer already familiar with microcontroller systems and is not generally suitable as the primary text for a microcontroller systems course. While this seems to be a drawback, observation of how most students use microcontrollers texts indicates it may not be very significant. If the concepts are sufficiently explained in class sessions, most students use the text primarily as a reference to provide examples and details on syntax and operation of the microcontroller development system. These, though, may be easily provided through using reference materials provided at no cost by Cypress, augmented by instructor developed examples and materials. Because of this, it is quite reasonable to operate the course without a textbook. This has the added advantage that is saves the students a significant expense. It is even possible to have students each purchase a CY3210-PsoCEval1 system of their own for less that half the cost of a textbook, saving the students money at the same time they gain a development system of their own.

While the discussion of the section has concentrated on specifics of the PSoC that require special solutions, the vast majority of microcontroller systems course objectives can be attained without any special consideration. Overall, the PSoC is very well-suited to a smooth running and successful microcontrollers systems course.

Assessing student preparedness for system design

At Seattle Pacific University, the microcontroller system design course is an important prerequisite to the capstone senior design course for electrical and computer engineering students. In the senior design course, student teams design and implement mixed-signal systems of their own conception. Most student teams choose to use at least one microcontroller in the system. In 2005-2006, the microcontroller systems design course at Seattle Pacific University was changed from a curriculum based on the aging Motorola 68HC11 to one based on the

Cypress PSoC. Students taking senior design in 2005-2006 participated in the 68HC11-based microcontrollers course, while students taking the senior design course in 2006-2007 received the PSoC-based microcontrollers course. By comparing student readiness to design, develop and test the microcontroller portion of their senior design projects between 2005-2006 and 2006-2007, some conclusions about the usefulness of changing the curriculum may be drawn.

The most significant difference between the ’05-’06 seniors and the ’06-’07 seniors is how many teams chose to use the microcontroller system they studied in the microcontroller systems course. In 2005-2006, all students were trained in using the 68HC11, yet no students chose to use it. This is consistent with the general conception that the lifetime of the 68HC11 has ended. Of the four teams in 2005-2006, three chose to use PSoC microcontrollers, while one chose to use a Microchip PIC microcontroller. The high number of PSoC projects can be attributed to the past success of previous seniors – word had been spread around that PSoCs were great for developing microcontroller systems. In these cases, the students trained themselves on the PSoC and PIC microcontrollers using company-supplied materials. In 2006-2007, three of the four teams chose to use PSoCs, which they had received previous training in. The other team chose to use the Texas Instruments MSP-430. While the sample size (four teams from each year) is small, it is still possible to make some general comparisons about the preparedness of the students in the senior design class. Two important areas are readiness (the ability to develop systems with limited help) and design quality.

The teams who took the PSoC-based course were more ready to develop systems than those who took the HC11-based course. The types of questions asked of the instructor tell most of the story in this case. Those who took the HC11 course frequently asked for help installing the development software, connecting the in-circuit emulator to their prototypes, and getting started programming. These students, who used PSoCs and PICs in senior design, caught on quickly enough through training and other reference materials, but first went through a period in which they were very timid about their designs and required frequent assistance. In contrast, those who took the PSoC-based course were much more ready to develop projects in senior design. Most had their development systems and in-circuit emulators running with no assistance. Their first questions tended to be advanced issues, such as dealing with interrupts and allocating resources. While much of the readiness can be attributed to their choice of the same microcontroller they had been trained in (the PSoC), even the team which chose the MSP-430 was able to make significant progress with very little help.

Students who took the PSoC-based microcontroller systems course also produced designs with fewer microcontroller-related flaws. In prior years, a significant portion of the projects had such flaws. For example, a team might choose a version of a microcontroller with too few I/O pins, or an incorrect supply or I/O voltage. Likewise, they might underestimate the processing power or memory space needed for their application. In the ’06-’07 course, projects exhibited these flaws at a significantly lower rate. This is most likely because many students chose to use the PSoC, in which they were experienced, and they were able to relate the concepts of the PSoC-based course directly to other modern microcontrollers. Overall, the small sampling of student teams has shown an association between the PSoC-based course and higher quality system designs in the senior design course.

Conclusion

Because success in microcontroller system design course can be very important for many electrical and computer engineering students, careful consideration of the microcontroller system chosen for the course is important. The Cypress PSoC provides an excellent platform for a number of reasons. It is an inexpensive, but well-rounded system. Its configurability makes it easy for students to experiment with a wide range of hardware and software components and an especially good platform for understanding mixed-signal systems. It has reasonable market share and is successfully used in a wide range of industrial applications. The support tools and evaluation systems are well-designed and well-maintained. Cypress Semiconductor provides a high level of support for the PSoC and is generous in donating hardware, software and support for educational purposes. Like any microcontroller system, there are some areas that don't fit perfectly when matching the PSoC with a particular course structure. However, there are reasonable solutions to all of the difficult areas. Students seem to enjoy the process of developing PSoC-based systems and have developed many successful designs based on the PSoC. Overall, converting a microcontroller system design course to use the Cypress PSoC can well be considered an upgrade.

Bibliography

1. M. Hedley and S. Barrie, "An undergraduate microcontroller systems laboratory," IEEE Trans. Educ., vol. 41, Electronic Distribution, Nov. 1998.
2. Manjikian, N. Simmons, S. "Evolution and Enhancements of a Microprocessor Systems Course," IEEE Trans. Educ., vol 42, Electronic Distribution, Nov. 1999.
3. Ashby, Robert. Designer's Guide to the Cypress PSoC, Newnes, 2005.