Embedded Systems in an Engineering Science Curriculum

Kevin Nickels and Matt Sealey
Trinity University, knickels@trinity.edu,
Trinity University and Genesi USA, msealey@trinity.edu, matt@genesi-usa.com

***Abstract* – The Embedded Systems Portability project investigates how to acquaint students in an engineering science curriculum with several important modern trends and practices in embedded system design. An overview of the program is given, and the specific objectives of several elective courses in electrical engineering are presented. The project also aims to provide a flexible but powerful controller for use in designs, most notably the open-ended senior design course. As these students are largely not deeply disciplinary, the trends of modular programming and functional portability need to be presented while minimizing prerequisite knowledge and skillsets. An embedded System on a Chip utilizing an ARM or a Power Architecture core is connected to a PCI-connected Complex Programmable Logic Device to provide computation power, flexible input/output, and a high bandwidth connection between the two. This powerful combination is connected to one of three representative plants for control. Example projects illustrating the portions of the system under study in several courses at Trinity are described.**

Index Terms – Embedded Systems, Complex Programmable Logic Device, Digital Logic Design

BACKGROUND

Trinity University is a small private liberal arts and sciences university in San Antonio, Texas. The Department of Engineering Science at Trinity University offers a broad-based engineering degree covering the fundamentals of electrical, chemical, and mechanical engineering. In the junior and senior years, students specialize in one of these fields. More detailed information on the program can be found in [1] .

The "fundamentals" electronics courses required of all engineering majors are Electric Circuits & Laboratory and Electronics I & Laboratory. These courses are taught every year and have primarily sophomore students. In these courses, the students are prepared for whichever electrically-oriented electives they encounter.

Due primarily to the small size of the department (25-30 majors/year with interests in EE, ME, and ChemE), most elective courses are offered in alternating years, and typically are taken by a mix of juniors and seniors.

Table I shows the progression of courses that might be taken by students with interests in electrical engineering. Each row of the table tracks a single class of students. Focussing on the class of 2011, in the sophomore year (08-09) they take Electric Circuits and Electronics I. In their junior year (09-10), they may take Mechatronics, along with the class of 2010, who are seniors at the time. In their senior year (10-11), they may take Digital Logic Design and Lab in the fall, and Embedded Microcomputer Systems in the spring, along with the class of 2012, who are juniors.

TABLE I
SEQUENCING OF COURSEWORK

Class of	Fall 2008	Spring 2009	Fall 2009	Spring 2010	Fall 2010	Spring 2011
2009	4365	4369				
2010	4365	4369		4367		
2011	2320	2364		4367	4365	4369
2012			2320	2364	4365	4369

Required Classes	Elective Classes
2320 – Electric Circuits	4365 – Digital Logic Design
2364 – Electronics I	4369 – Embedded Micro. Systems
	4367 - Mechatronics

The class of 2010 or 2012, in contrast, will take the (same) elective courses in a different order. This nonlinearity of courses forces careful consideration of the independence of modules in these courses.

Engineering students must learn to program in a high-level language such as C, C++, or Java, but this is a graduation requirement and thus may not be set as a prerequisite for any of these courses.

Thus, for a given elective course, one student may have had single year of introductory circuits and electronics, while another may have had three other electrical/computer oriented courses and be proficient in C. Of course, other programs also struggle with challenges in course design due to students with diversity in backgrounds [2]

In this environment, one course in Digital Logic Design (DLD) is offered, along with a supporting laboratory course. This contrasts to a traditional electrical or computer engineering program, where this same topical coverage would be handled in more depth in two or even three

semesters. Objectives for this course include basic combinational and sequential logic, leading up to a case study of a complex sequential machine – recently a simple five-instruction computer. In the laboratory, supporting projects are implemented on Altera complex programmable logic devices (CPLDs), using VHDL throughout [4] .

Embedded Microcomputer Systems currently utilizes Motorola 6811 evaluation boards to teach assembly language, embedded controller architecture and modular programming. While there is not a separate supporting laboratory course, many of the weekly problem sets contain significant laboratory work. Objectives for this course include embedded system architecture, systematic debugging, use of the stack and hardware control/status registers, interrupts, timer/counters, and interfacing techniques from the processor's point of view.

Mechatronics is a relatively new course in the department. It replaced a modern control theory course, and focuses on systems involving computer control, electric and electronic instrumentation and actuation, and system models. The heart of the course is the study of the tradeoffs between mechanical design of a system, the system model for that system, and the control of that system model.

Senior Design is the capstone design course, the culmination of eight required semesters of design at Trinity. Students in this course work in small groups (4-6) with a faculty member on a yearlong design project. Robotic and mechatronic projects are popular due to their inherent interdisciplinarity. The controllers for these projects often reflect those used in the elective courses described above, as that is what (some) students are familiar with. In recent years, general-purpose PCs have been increasingly used with USB/GPIO, USB/I2C, or similar peripherals in place of a classical microcontroller. Primarily, this is due to the designers' desire to include complex tasks such as facial recognition, navigation, etc. that would be extremely challenging to embed in a microcontroller.

Genesi USA, Inc., established in 2003 and based in San Antonio since 2004 and with a research and development division in Frankfurt, Germany, develops affordable, low-power embedded systems designs and an IEEE 1275 "Open Firmware" BIOS implementation with hardware abstraction and virtualization capabilities.

Since Spring 2007, Genesi and Trinity have been collaborating on utilizing the EFIKA 5200B evaluation board, based on the Freescale MPC5200B Power Architecture™ System-on-a-Chip (SoC) in design and research projects. It was used in 2007-2008 on a Tree Climbing Robot and Line-Following Robot design project, and is being utilized for an office navigation robot.

PROJECT OBJECTIVES

The Embedded Systems Portability project targets engineering science students in several elective courses with different topical coverage. The project aims to acquaint students with several important modern trends and practices in embedded system design – those of modular programming and functional portability – while minimizing prerequisite knowledge. The project also aims to familiarize students with modern embedded SoCs such as those based on ARM or Power Architecture cores instead of utilizing the easily understood but dated "classical" 8-bit non-pipelined 6811 microcontroller.

This will result in several pedagogically relevant systems illustrating these trends in ways accessible to our general engineering students.

Another objective of the project is to develop a flexible system for computer control of various electro-mechanical design projects. The ESP architecture has the ability to include software computation, off-the-shelf peripherals (Linux drivers allowing), hardware computation, and custom and flexible sensor processing in one compact package. The only discrete logic required for these electro-mechanical designs involves voltage interfacing and current amplification.

FPGAS AND CPLDS

Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) are architecturally different chips that achieve the same overall systems-level objective. They allow digital designs to be developed in either graphical or HDL (Hardware Description Language) software on a PC and field programmed by the user. This speeds up development times with respect to either discrete chips on a printed circuit board or the design of a fully-custom application specific integrated chip.

Altera markets a wide variety of FPGAs and CPLDs that are available on several different development boards. The UP2 and DE2 boards, for example, feature the MAXII CPLD and the Cyclone FPGA chips along with programming support hardware, easily accessible peripherals such as switches and LEDs, and popular interface connectors such as VGA, audio in/out, etc. One tradeoff in choosing a development board is that the connectors and peripherals on the board consume pin resources on the CPLD and board and detract from the amount of General Purpose Input/Output (GPIO) available. This must be balanced against the convenience of using a predesigned, debugged board for the prototyping of projects.

The UP2 boards are used at Trinity in courses including Engineering Design V, taken by all engineering majors, and Digital Logic Design, an electrical engineering elective.

Several Altera chips are also available on PCI-based (Peripheral Component Interconnect) development boards, which are similar but include a standard PCI connector for

high-bandwidth communication to a computer such as a PC, or in our case, an EFIKA.

THE EFIKA AS A PLATFORM

The EFIKA, shown in Figure 1, was designed as an easy-to-use evaluation board for the Freescale MPC5200B SoC implementing a 400MHz CPU core and very high integration of peripherals[6] . Expansion connectors – such as USB, audio and PCI bus slot – have been exposed from the SoC in order to speed development and adoption of the processor as a working device for software developers. This is in contrast to the rather minimalistic reference designs usually provided by silicon vendors.

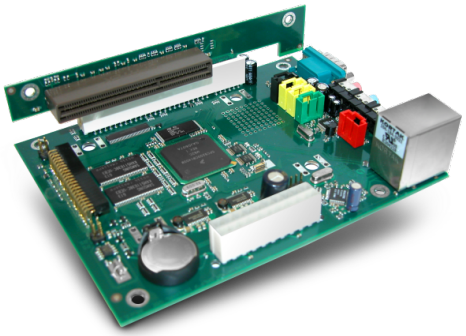

FIGURE 1
THE EFIKA MOTHERBOARD

It would be possible to combine the CPU and custom logic using a synthesized CPU core such as Altera Nios II [7] , however this would overcomplicate the design of the CPLD portion and would require the use of a (much) more expensive CPLD. Since the EFIKA offers a fixed CPU and a wealth of peripheral exposure, these need not be implemented on the CPLD, increasing the resources available and simplifying the development of student designs.

The EFIKA configuration used in the project was simply a bare motherboard with the addition of a 2.5" hard disk drive for storage, and housed in an industrial steel case with an Altera MAXII PCI CPLD connected, as shown in Figure 2.

The most important expansion opportunity with regards to the Embedded Systems Portability project is the PCI bus slot, into which the CPLD development kit can be inserted.

ESP ARCHITECTURE

The basic architecture of the system is to connect a standard embedded CPU such as Power Architecture or ARM to a CPLD, which is then connected to certain "plant" devices containing motors and sensors. This basic layout is illustrated in Figure 3.

On the EFIKA, PCI was chosen as the interface bus for convenience since it is the most easily accessible high bandwidth bus. The architecture can use any bus that presents a memory-mapped view of peripherals (e.g. local processor buses such as AXI on ARM) with very little change in the software operation, and very little change in the design of the "glue logic". The term "glue logic" refers to HDL modules that link the interface bus to custom peripherals. For PCI this includes the Altera PCI Megacore IP and PCI "target" control logic as a finite state machine generating signals for address, read/write data bus, write enable signals which are passed to all control and measurement modules.

FIGURE 2
EFIKA SYSTEM WITH PCI CPLD DEVELOPMENT BOARD

There are several off-the-shelf technologies that perform this task as IP cores such as the Altera Avalon bus. However it was deemed too complex to implement in view of the resources available on the CPLD. A significant learning curve would also have proved difficult to justify imposing on the students through the planned coursework.

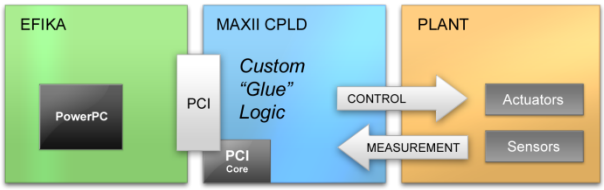

FIGURE 3
ARCHITECTURE OVERVIEW

I. Plants

Several plants were chosen as targets for this project. The term *plant* is used in control theory to represent the physical process being controlled as well as the actuator(s) used to implement the control. In this paper, we expand the definition to also include measurement sensors.

The first, and simplest, plant is a Revell Visible V8 Model Engine. The hand crank has been replaced with a DC gearmotor to drive the engine, and a line following sensor has been installed on the flywheel to sense the speed of the engine. This is a simple plant, so unidirectional velocity control can be achieved with a proportional controller.

The second plant chosen is a five-axis toy robotic arm (OWI535 or OWI007). DC motors (in turn driven by H-

bridge circuits) drive this arm. A mixture of line detectors and optical shaft encoders achieves position feedback for each axis. Control of this plant is made more complicated by the bidirectional control needed and by the five motors that need to be controlled.

The final plant chosen to control is actually an interface specification rather than a physical plant. In the undergraduate controls laboratory, Trinity has a variety of different plants for students to use in their introductory control class. These very "classical" plants, such as a ball and beam, coupled-tanks, or ball and hoop, take a -10 to +10VDC input voltage[1] for the actuator and generate a -10 to +10VDC output voltage from the sensor.

This set of plants represents an array of actuators, measurements, and control styles. If a flexible controller is developed that can interface easily to each of these plants, it should generalize well to other projects that Trinity's students with to implement in their design courses as well as the targeted experiments described in this paper.

II. CPLDs

The Altera MAXII PCI CPLD development kit was chosen primarily because of its affordability and expansion potential. The MAXII CPLD on the card provides 1270 "Logic Units" (LUs) of which the PCI Interface IP-Core takes up some 50% of available resources. This leaves around 600 LUs for student designs.

The development board also provides 41 GPIO pins and several useful components such as SRAM, LCD panel, LEDs, and switches. Many of the connectors and interfaces (such as USB, audio, VGA, etc) are not needed for our project, since students will use an EFIKA that has these connectors.

The CPLD is programmed with digital logic to handle bus protocols (in this case, PCI) and present this as simple address, data and control (interrupts, data direction) to the plant controller modules. This digital logic, referred to in Figure 3 as "glue logic", may be a black-box IP core component (such as the Altera PCI Megacore IP), or may be another module designed for the students. In any case, the students do not know – or care – about the internal operation.

Beside this module is another black-box module that interfaces the demultiplexed bus core to a set of standard registers. These hardware registers (sets of D flip-flops to the students) provide the interface between the VHDL circuit and the software on the CPU.

The VHDL circuit uses write-only registers[2] to set parameters and control logic, and passes information from the controller and sensors back to read-only registers. The control circuit can be as simple as passing a software-generated PWM duty cycle or as complex as a full PID controller. The sensor processing circuit will often contain hardware computations such as counters for encoders. The incredible flexibility afforded by a CPLD able to implement arbitrary digital logic is intended to replace many commonly-seen discrete controllers such as USB/I2C or USB/GPIO controllers in student designs.

This architecture also allows high visibility of the operation of the CPLD-based design. Since every exposed register is present in a linearly mapped memory area to the CPU, contents of registers can be "dumped" in real-time on the CPU to present the current state of the controllers. The true benefit is the ability to read and write registers with ordinary CPU memory read and write instructions, and present these as human-readable values, or to perform calculations and control loops.

III. Embedded CPU

Linux was chosen for the CPU because of its low cost and availability. Students are also more likely to have some exposure to Linux compared to proprietary real-time operating systems (RTOSs) such as QNX or VxWorks. Proprietary RTOSs may also cause the university to incur substantial licensing fees.

The Linux kernel driver offers the ability to expose the design – for instance the contents of a register – as a file through kernel filesystems, as illustrated in Figure 4. As an example, the driver may expose a file at location "/proc/trinity/setpoint". A value written to this file is passed to the kernel driver that then parses and converts it to a write to the PCI registers, or it may simply modify the behavior of a software control loop. Read operations may also be used to show the results of the control effort – implemented in the software driver or in the CPLD logic – in a human-readable format.

However, this method gives no real fine-grained control over the design itself from a user-space program. Visibility of the design from user-space is lower than that of the design from inside the kernel driver. Since it is difficult to signal interrupts to user-space, the entirety of any time-sensitive control for the design must be implemented inside the kernel driver when using traditional methods.

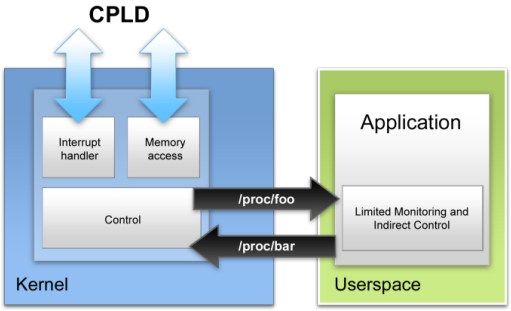

FIGURE 4
KERNEL DRIVER WITH FILESYSTEM INTERFACE

To work around this, we opted to use the recently implemented Userspace IO (UIO) kernel framework, illustrated in Figure 5. This allows exposing kernel features such as interrupt signaling and physical memory mapping to a user-space program enabling full control of the PCI design

[1] Unidirectional plants use 0-+10VDC for their respective input/output
[2] From the point of view of the CPU – they're read-only to the CPLD.

in an application using the same CPU read and write instructions as in the kernel driver.

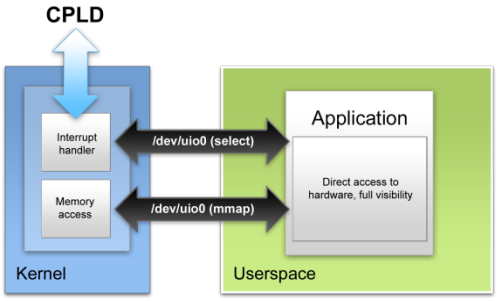

FIGURE 5
UIO KERNEL FRAMEWORK ALLOWING DIRECT HARDWARE ACCESS FROM USER APPLICATION

The flexibility of the UIO framework enables the use of a single kernel driver module for every student project, reducing their exposure to kernel subsystems. The initialization of the user-space application can be sufficiently abstracted as an external code module which can be linked to the student's own code, effectively removing the need for the student to become an expert in Linux kernel or application programming and instead concentrate on implementing control and measurement.

TASK ALLOCATION

Each control system contains the same basic components. As an example, consider the system shown in Figure 6. With the ESP architecture described above, several components of the system can be moved between the CPU and the CPLD, depending on the pedagogic needs of the experiment.

For example, a digital logic experiment may use the CPU only to set a setpoint (v_r in Figure 6), and have the CPLD circuit subtract the measurement (v_m) and compute the control effort (u). Another CPLD circuit takes the sensor input (y) and processes it to obtain the measurement (v_m).

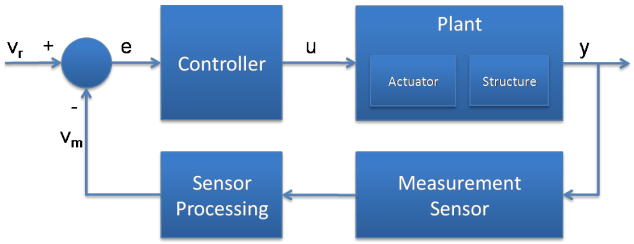

FIGURE 6
CONTROL SYSTEM

An experiment in a mechatronics course, however, may wish to investigate different control strategies. Thus, the VHDL circuit may use the interface registers only to copy the control effort (u) to the actuator, and do minimal or no sensor processing. The controller is then implemented entirely in the CPU.

This ability to move tasks between "hardware design" in the CPLD and "software design" in the CPU allows the same hardware architecture to be used for experiments in several different courses, as well as for designs.

COURSE OBJECTIVES AND COURSEWORK

Digital Logic Design (DLD) has objectives encompassing combinatorial and sequential design, and the attached laboratory additionally has goals including utilization of Hardware Description Languages and modern programmable logic.

In order to support these objectives without requiring background in programming, computer architecture, or embedded systems, the focus shifts to the CPLD portion of the system. Complete software drivers, including tutorial-style documentation, will be provided to the students. Thus, the CPU interface seen by the students' designs consists of a set of hardware registers (D flip-flops). A black-box module controls these registers, independently handling bus interactions and demultiplexing, and other topics outside the scope of the course and experience of the students.

Labs envisioned for DLD include implementation of custom interface logic controlled by these hardware registers. For example, students could be asked to generate custom pulse width modulation (PWM) waveform generators based on a PERIOD and HITIME register set by the software, or to have a counter indexed by a shaft encoder, that sets a hardware register to be subsequently read by the CPU.

Some important concepts to the embedded microcomputer systems class include hardware control and status registers and how to interface to them, and modern practices in embedded systems development.

Designs that may support these objectives include those referenced above for DLD, including e.g. PWM generators and encoder counters, but as students cannot be assumed to have had DLD, an instructor-generated VHDL implementation will be provided to this class. Likewise, kernel driver design is outside the scope of the course, so a full implementation of the kernel drivers will be provided.

Embedded students will be provided a skeleton UIO framework that shows how to read and to write a hardware register in C. They may then be asked to implement, say, a PID controller around this framework. Since the students cannot be assumed to know C programming, tutorial-style directions for compiling and running the programs will need to be developed, along with some intermediate checkpoints to evolve a working control scheme.

Mechatronic design of systems, with its simultaneous attention to mechanical, electrical, and control, provides another application area for these systems. These students would concentrate on the various control architectures represented by the systems.

Specific course objectives for mechatronics include the theory of digital control and the z transform, the ability to use a variety of control architectures in the control of

systems, and exploiting several different sensors/actuators to implement a mechatronic system that carries out a specified objective.

In addition to the applied nature of the lab projects in mechatronics that would benefit from a highly capable system such as the ESP architecture, these students would benefit from a study of the portability of control between the CPLD and the CPU. In order to address the limited technical background of the students, modules would be provided for a simple and complex CPU controller, and for a simple and complex CPLD controller. The study of the tradeoffs between implementing complex control (e.g. PID control) in hardware, in VHDL, and in pure software, could take advantage of this architecture as well.

Senior Design is a year-long capstone, with objectives centered (within the scope of this paper) around utilizing the architecture as a controller. Unlike the disciplinary electives discussed above, design groups are formed with necessary skill-sets in mind. Thus, at least one member of a group utilizing this controller is likely to know C programming, or VHDL programming, or many of the other topics needed.

With the example and demonstration modules and the skeleton designs discussed above, design groups should be able to utilize this architecture to command and control custom hardware in a very short development cycle. Typically, the fall is devoted to classical design and parts specifications, and the spring is used to build, test, and debug the designs. Since designing and implementing a custom CPU/CPLD printed circuit board is outside the scope, in engineering time, expense, and experience, of senior design, the ESP architecture would benefit the design students greatly.

The students of engineering science at Trinity are in a unique position with respect to embedded systems design and practice. They may study these systems from several viewpoints, but approach them from a generalist perspective. This project looks at ways to acquaint Trinity students with concepts in embedded systems portability using modern methods and up to date hardware.

STATUS & FUTURE WORK

This paper describes the ESP architecture, comprised of an embedded CPU, a PCI-connected CPLD, and programming on each. A prototype plant (the V8 engine) has been interfaced to this controller, and a simple control system (proportional controller) has been implemented on the CPU. This prototype uses Altera's PCI IP-Core and some custom VHDL to interface the CPLD design to the CPU. The CPLD design contains, in addition to the interface logic, PWM pulse wave generators and counters to read the line counters.

The second target plant, the five-axis arm, is being interfaced to the system at the time of writing. The physical and electronic modifications required to interface this arm to the system are underway.

Once all three plants have been interfaced, and several different control schemes have been implemented on each, laboratory experiments utilizing this platform and plants will be developed. These will replaces the complete reference design with student designs that lack one component (that under study). So a digital logic design experiment using this system might ask the student to implement only the PWM and sensor processing portions of the V8 interface, while a mechatronics experiment might look at different driver circuits that could be used for the gear motor or different control algorithms on the CPU.

Similarly, the full reference designs will be provided to Senior Design students for use as a design component in their projects. This will enable students to utilize sophisticated modern embedded practices into their projects without designing and implementing their own custom boards.

ACKNOWLEDGMENT

The authors gratefully acknowledge Genesi, Altera, and Freescale for their monetary and equipment support of the Embedded Systems Development Lab at Trinity. Thanks also go to André Siegel of Genesi for his help in producing diagrams for this paper.

REFERENCES

[1] Uddin, M. "Multidisciplinary Engineering Science program at Trinity University". *Proceedings American Society for Engineering Education Annual Conference*, 2004.

[2] Stephan, K. and Sriraman, V. "A Digital Electronics Course Using CPLDs for Manufacturing Engineers." *Proceedings American Society for Engineering Education Annual Conference*, 2004.

[3] Hamblen, J.O. "Using a Low-Cost SoC Computer and a Commercial RTOS in an Embedded Systems Design Course." *IEEE Transactions on Education*, Vol 51, No. 2., 2008

[4] Nickels, K. "What are the "Fundamentals" of Modern Digital Logic Design?" Proceedings American Society for Engineering Education Gulf-Southwest Annual Conference, 2005.

[5] Hamblen, J. O., Hall, T. S., and Furman, M, D. "Rapid Prototyping of Digital Systems Quartus II Edition. Springer, 2006.

[6] Genesi U.S.A., Inc, "EFIKA Micro-Motherboard." [Online document] [2009 Mar 23] Available at http://www.genesi-usa.com/efika

[7] Hall, T.S. and Hamblen, J.O. "System-on-a-Programmable-Chip Development Platforms in the Classroom." *IEEE Transactions on Education*, Vol 47, No. 4., 2004

AUTHOR INFORMATION

Kevin Nickels, Associate Professor, Engineering Science, Trinity University, knickels@trinity.edu.

Matt Sealey, Product Development Analyst, Genesi USA, Visiting Scholar, Trinity University, matt@genesi-usa.com, msealey@trinity.edu.