An Embedded Systems Course and Course Sequence

Kenneth G. Ricks*
Electrical and Computer Engineering
The University of Alabama
Tuscaloosa, AL 35487, USA
kricks@coe.eng.ua.edu
* Contact Author

William A. Stapleton
Electrical and Computer Engineering
The University of Alabama
Tuscaloosa, AL 35487, USA

D. Jeff Jackson
Electrical and Computer Engineering
The University of Alabama
Tuscaloosa, AL 35487, USA

Abstract

Recently, the University of Alabama Department of Electrical and Computer Engineering adopted curricular changes to incorporate embedded systems into its computer engineering core course sequence. One of the major changes implemented was the creation of a senior lecture/laboratory combination specifically dedicated to embedded systems. This paper describes the specific lecture and laboratory content of this senior-level course and how this course fits within the new curriculum a The University of Alabama.

1. Background/Introduction

The faculty of the Computer Engineering program at The University of Alabama has undertaken a project of pedagogical improvement by incorporating a focus on embedded systems that is pervasive throughout the computer engineering curriculum. There are several driving factors behind this decision. Embedded systems represent a major fraction of the digital systems market as indicated by the fact that embedded systems represent a key technology in the automotive, consumer electronics, industrial automation, military and aerospace applications, office automation, telecommunication and data-communication industries [1-3]. There is also significant regional interest in embedded systems with several major automotive and other manufacturing industries located in the state of Alabama and surrounding areas [4].

As much as 98% of all 32-bit microprocessors currently in use worldwide are used in embedded systems [5]. However, most computer engineering programs teach programming and design skills that are appropriate for a general-purpose computer operating under control of a commercial operating system rather than for the more specialized embedded systems [6]. Additionally, instruction in embedded systems can increase opportunities for breadth in a curriculum as these systems naturally involve hardware and software components that interface to various electrical, mechanical, and chemical processes. Thus embedded systems education is an excellent example of an area of study that requires depth and rigor while maintaining breadth required for meeting emerging workforce and education needs of U.S. industry [4, 7].

The rapid proliferation of embedded systems requires an increasing number of engineers trained in microcontroller-based systems, real-time concepts, hardware/software co-design, distributed processing, hardware/software integration, and system-level issues in embedded systems design. Instructional material is just beginning to appear in this area and the development of this focus area, associated instructional materials, and evaluation materials will allow us to better serve our students and, more importantly, to provide material for this emerging area that can be adapted for use by others.

This embedded systems focus is important in the context of distinguishing our programs at The University of Alabama. The embedded systems focus will directly affect three degree programs: Computer Engineering, Computer Science, and Electrical Engineering. The majority of computer engineering programs deal primarily with design and programming for general-purpose computers. Traditionally, we also have offered a broad exposure to computer engineering topics in our curriculum and conducted research in a number of areas. Recent self-assessments of our program utilizing both the IEEE/ACM model computer engineering curriculum [8] and a set of nationally recognized and comparable programs led us to choose to adopt a more focused curriculum model. Because of our limited size and resources, we believe that focusing both our education and research efforts on a single theme, namely embedded systems, will allow us to progress in both areas. A web-based search for “embedded systems education” using the ASEE database and internet search engines reveals a scarcity of programs focusing on embedded systems, particularly in the U.S. Southeastern region. We believe that successful implementation of this focused effort in a niche area will serve as a model for many other similarly sized programs [4].

2. The University of Alabama Computer Engineering Core Course Sequence

The plan for reforming the curriculum will involve each of the courses in the Computer Engineering program shown in Figure 1. In this figure, the arrows denote a prerequisite relationship between the courses. The comprehensive plan builds upon each of these courses to provide an enriched experience for the students.

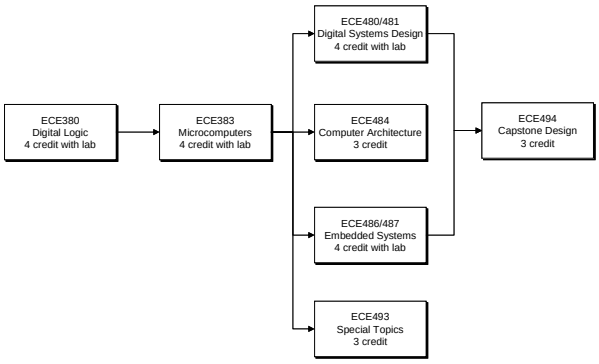

Figure 1. Computer Engineering Core Curriculum with Embedded Systems Focus

The first course in the sequence of Figure 1, ECE 380 - Digital Logic, is a four-hour lecture/laboratory combination class incorporating traditional combinational and sequential logic design and digital design using VHDL. The embedded systems theme is incorporated into this class through exercises that, for example, include digital counter designs in the context of watchdog timers common in embedded processors, pulse width modulation (PWM) circuit design, and complex state machine designs for typical embedded system tasks such as bus arbitration. Altera's Quartus II electronic design automation software is used to provide the students with system design and simulation experience. This course is required for students in all three directly affected engineering disciplines. Along with the nature of the subject material, this student diversity makes this course especially well-suited for the incorporation of multidisciplinary team-based learning. Finally, basic designs from exercises in this course are used as components in larger, more complex designs in subsequent courses. Proper design techniques as well as design reuse are stressed.

The second course, ECE 383 – Microcomputers, builds on a foundation of traditional architectural topics such as register, memory, bus, and instruction set design to incorporate embedded systems topics such as peripheral interfacing, analog-to-digital (A/D) conversion, device control, interrupt management, and system reliability. Metrowerks CodeWarrior is used to provide a modern development environment for programming and debugging the software portions of system design. Students expand the use of Altera's Quartus II software introduced in ECE 380 to produce custom interface logic to connect a microprocessor with a variety of peripheral devices. We also introduce the basic use of Mentor Graphics software for facilitating hardware/software co-design and board-level design issues. As with ECE 380, this course is required for students in all three directly affected engineering disciplines facilitating the incorporation of multidisciplinary team-based learning.

The third course, ECE 480/481 - Digital Systems Design, is a four hour lecture/laboratory combination class that focuses on the design and test of digital systems components including basic arithmetic and logic components, and digital systems interfaces including PWM designs, and mouse, keyboard and video display drivers. VHDL-based designs are implemented on FPGA devices. System-on-a-Programmable-Chip design methodologies are introduced. Special emphasis on testing includes an introduction to device-embedded logic analyzers and their use for debugging SoPC designs. Specific topical material introduced includes hardware description languages, electronic design automation, logic circuit testing and testable design, SOC design and intellectual property (IP) cores. Software tools for electronic design automation from Altera and Mentor Graphics corporations are used, allowing students previously exposed to these toolsets to become more proficient in their use. More advanced features of these toolsets are introduced including floor planning, advanced timing analysis, and synthesis options. Additional toolsets are introduced including both design-for-test and hardware/software co-design for embedded processors. Additionally, the Mentor Graphics toolset includes capabilities for engineering project management that are used to manage the execution of best design practices throughout project assignments. Specific embedded systems concepts that are covered include embedded processor design, peripheral integration and SOC solutions for embedded systems. Integration of custom hardware and software with existing components is emphasized. Hardware/software co-design is addressed by integrating and expanding basic projects from the first two courses: ECE 380 and ECE 383 [4].

The fourth course, ECE 484 – Computer Architecture, is a three hour course that incorporates embedded systems concepts into the context of computer architectural issues. Traditional computing architectures are introduced, evaluated, and contrasted with embedded systems architectures [9]. Specifically, architectural design tradeoffs associated with the processor(s), input/output (I/O), and memory are

discussed. Performance evaluation and analysis is also contrasted between a general-purpose MIPS architecture and architectures used in embedded systems. Hardware/software co-design is introduced, and the relationships between the software and hardware components of computing systems are discussed.

The fifth course, ECE 486/487 – Embedded Systems, is a four hour lecture/laboratory combination class. It is described in detail in the following sections of this paper.

The sixth course, ECE 493 – Special Topics, provides flexibility in the curriculum by allowing advanced embedded systems concepts to be introduced on a regular as-needed basis. Such topics would include, but are not limited to, real-time systems, distributed embedded systems, hardware/software co-design methodologies and design verification/validation/testing.

The seventh course, ECE 494 – Capstone Design, culminates the undergraduate engineering design experience by providing a semester-long, team-oriented design project building on the skills learned in a previous senior-level lecture/laboratory course. Candidate lecture/laboratory courses preceding the Capstone Design course include ECE 480/481 Digital Systems Design and ECE 486/487 Embedded Systems. All facets of the previously introduced software tools will be exercised in this course. Design projects such as programmable logic devices and SOC solutions in robotic car competitions [10] and projects following the IEEE Computer Society International Design Competition model [11] will be used. Since the design is team oriented, this course also provides the opportunity to assess student teaming skills and the pedagogies used throughout the curriculum for instruction in teaming [4].

3. ECE 486/487 Embedded Systems

The ECE 486/487 Embedded Systems lecture/laboratory course is a new course resulting from the curriculum reform activities. The following sections describe the concepts covered in the lecture, how these concepts relate to the IEEE/ACM model curriculum, the laboratory activities, and the hardware and software currently used for the laboratory assignments.

3.1 Lecture Material

The course begins with an introduction to embedded systems. This portion of the lecture provides general definitions of embedded systems, examples of common embedded systems, and distinguishes embedded systems from other types of computing systems. Also, general characteristics of embedded systems are given and functional and non-functional metrics used to evaluate system design and performance are described. Background material such as Moore's Law is presented to explain the broad emergence of embedded systems throughout our society. This leads to a justification of embedded systems as a focus area within computer engineering and the corresponding need for embedded systems education. This material corresponds to various core components of the IEEE/ACM model curriculum including "History and overview of embedded systems – ESY0" and "Classification of embedded systems – ESY6", as well as one elective component of the model called "Software engineering considerations – ESY7".

The next set of lectures is designed to concentrate on the design of embedded systems. Specifically, ad-hoc, top-down, and bottom-up design methodologies are shown to be inadequate as general-purpose methodologies due to the varying system requirements and characteristics across multiple embedded systems applications. Hardware/software co-design is introduced and compared to the other methodologies. Its uses a domain-independent process abstraction to describe system behavior which delays hardware and software allocation and mapping decisions making it more suitable as a general-purpose approach for these applications. The main goals of this concept are that embedded systems designers must be able to perform hardware and software design tradeoffs and analysis. Computational models used to describe system behavior are also introduced. These lecture concepts correlate to several of the components in the "Software engineering considerations – ESY7" section of the model curriculum which is recommended as elective material [8].

The aforementioned lecture materials represent a high-level, abstract view of embedded systems. Some of these concepts, particularly the design methodologies, are difficult for students to grasp, and students have indicated that these sections of the lecture are their least favorite. The following sets of lectures deal with more tangible concepts that are more easily mapped to hands-on laboratory assignments. Students have indicated a higher level of interest in this material.

The next set of lectures is designed to discuss typical I/O activities and related concepts required of embedded systems. Specifically, data acquisition, A/D conversion, digital-to-analog (D/A) conversion, sampling rates, the Nyquist rule, A/D resolution, "system" resolution, PWM, timers, timer resolution, communication protocols, direct memory access, and specific I/O devices such as keypads, and UARTs are discussed. Many of these concepts are introduced earlier in the course sequence, but in this case a

concerted effort is made to put these concepts into a "system" context. For example, A/D conversion is introduced in ECE 383 in the context of an on-chip converter incorporated with the microprocessor. In ECE 486, A/D conversion is again discussed, but this time it is seen as part of a data acquisition system and the A/D converter is incorporated as an off-chip I/O peripheral device. In this case, the A/D converter resolution and sampling rate are compared to the requirements of the "system" within the context of the specific real-world data being collected. These topics are listed as components in two parts of the model curriculum including "Fundamentals of embedded systems – ESY1" (core) and "Hardware considerations – ESY3" (elective) [8].

The different architectures to support interfacing required for the I/O activities previously mentioned is the focus of another set of lectures. In particular, bus-based architectures are discussed and specific designs are created. Bus communication protocols are compared, master-slave relationships are defined, and system activities are decomposed into atomic bus transactions. Bus arbitration is introduced, multiprocessor bus architectures are described, and bus saturation is defined and explored. Finally, interrupt-driven and polled I/O are described, compared, and contrasted in terms of hardware design, software design, and system performance. All of these topics satisfy many of the components in the following parts of the model curriculum: "Language issues – ESY2" (core), "Hardware considerations – ESY3" (elective), "Mapping between languages and hardware – ESY4" (core), "Classification of embedded systems – ESY6" (core), "Particular techniques and applications – ESY8" (elective), and "High integrity software systems – ESY10" (elective) [8].

Another set of lectures is designed to address memory concepts. These lectures cover different memory technologies and discuss particular applications of each. The technologies are compared and contrasted based upon their operational characteristics. Also, memory system hierarchical design and caching are introduced. The localities of reference upon which memory system design is based are used to show the importance of memory system design and its effect on overall system performance. The particular aspect of the model curriculum incorporated into these lectures is "Mapping between languages and hardware – ESY4" (core) [8].

The last set of lectures is designed to introduce real-time issues. Real-time systems are defined and the various types are compared and contrasted. Real-time operating systems are discussed and their performance goals are described as they relate to I/O activities and memory operation addressed in earlier lectures. For example, at this point students seem to recognize and understand the effects of caching on real-time performance and the minimization of interrupt latency with real-time operating systems. The students have shown genuine excitement about being able to relate such concepts. Scheduling is also introduced at this point. Since we have already defined the process abstraction and the concurrent process model of computation, it is easy to address process scheduling, preemption, non-preemption, priority-based scheduling, and priority assignments based upon popular algorithms such as the rate-monotonic algorithm. These topics correlate to the following parts of the model curriculum: "Language issues – ESY2" (core), "Mapping between languages and hardware – ESY4" (core), "Real-time operating systems – ESY5" (elective), and "Classification of embedded systems – ESY6" (core) [8].

3.2 Laboratory Hardware and Software

The hardware and software dedicated to the embedded systems laboratory assignments uses a single-bus architecture built around the VMEbus. The VMEbus is a standardized bus protocol designed for I/O intensive operations and often used in industrial, military, and aerospace embedded applications [12]. Each of the three lab stations consists of two single-board-computers (SBC) connected to the VMEbus, one 6U-sized combination VMEbus CDROM drive and hard drive for each SBC, and one shared A/D board consisting of 64 differential analog input channels also connected to the VMEbus. One SBC is loaded with the Windows XP Professional operating system and the second SBC is loaded with Redhat Linux version 9.0 running the 2.4.20-6 Linux kernel. A customized library of software functions compatible with the C programming language is available for use on each platform. The functions make interfacing to the VMEbus address space easy and eliminate the need for timely driver development for the specific hardware used. Each of the three lab stations allows for remote login via the Internet. This promotes sharing of the hardware. Remote login does not provide for interacting directly with the equipment in some cases, for example setting up analog input into the A/D board. But, it does allow for software development which accounts for a majority of the time spent using the stations.

Although the VMEbus is seen almost exclusively in industrial, military, and aerospace applications, it is surprisingly useful for academic embedded systems activities. In addition to using its asynchronous protocol as an example of such bus communications, the flexibility of the VMEbus makes it perfect for demonstrating many other topics discussed in the IEEE/ACM model curriculum. For example, SBCs can

be easily added to a VMEbus backplane to produce a multiprocessor. The SBCs can be the same producing a homogenous multiprocessor, or each can be different, even executing different operating systems, to produce a heterogeneous multiprocessor. Various memory configurations can be set up by adding global memory cards to a VMEbus system. Multiprocessors and shared memory provide the opportunity to address mutual exclusion, concurrency, and inter-process communication issues. Various operating systems including real-time operating systems are readily available for VMEbus SBCs. With such an operating system, detailed timing analysis of system performance and real-time scheduling concepts can be investigated. The VMEbus supports various bus arbitration methods, has a prioritized 7-level interrupt protocol, supports multiple bus masters, has a data transfer rate of 40 Mbytes per second, and is standardized. Its thorough I/O support makes it easy to study polled I/O, interrupt-driven I/O, standard and memory-mapped I/O configurations, arbitration for multiple interrupting devices, starvation, and bus saturation concepts. One final benefit of the VMEbus is that there are many vendors and many choices for VMEbus devices making off-the-shelf components common, relatively inexpensive, and simple to use.

3.3 Laboratory Activities

The laboratory activities are chosen to supplement the lecture material. Each assignment is made with the goal of supporting the "system" concept of an embedded system. So, in each case, overall system performance is a concern. Based upon the data presented in [13], the C programming language is used for approximately 80% of all embedded systems, and assembly language is used for approximately 10%. Since assembly language is the choice for earlier courses in the UA sequence, such as ECE 383, this is the best time to introduce C as a high-level programming language suitable for embedded applications. By doing so, the laboratory addresses a core topic in the IEEE/ACM model curriculum called "Language Issues – ESY2". This specifically refers to a need for the description of various programming languages used in embedded systems and the specification of a guide for when such languages are appropriate [8]. Finally, each assignment will use the VMEbus systems described in the previous section or will involve a software simulation of some embedded systems component.

Another important aspect of the laboratory assignments is that the technical data necessary to program the hardware and to use the custom C software libraries is not presented in a formal fashion. Instead, students are responsible for gathering the necessary information from the technical documentation accompanying the laboratory hardware and software, i.e. technical manuals. This type of experience is invaluable to embedded systems engineers who will be faced with this task early and often in their careers, often dealing with documentation that is poorly written and filled with errors. Thus, the laboratory activities provide an opportunity to assess student learning in an unstructured environment.

The first two laboratory assignments involve the creation of a data acquisition system. The particular analog data collected from the real-world is not as much of a concern as how the data is collected and what is done with the data. For the first iteration of the course, the students collected environmental data including temperature, light, and humidity. The sensors and the circuitry required were pre-selected and set up for the students. This represents a case where practicing engineers are given an I/O component, i.e. a sensor package, with which to work and must integrate that package into the data acquisition system. In this way, the students can focus on system integration activities and avoid electronic design issues they should have been exposed to earlier in the curriculum and that tend to distract some students from the goal of the current exercise. For the first laboratory assignment, students create a data acquisition system that uses polled I/O to collect environmental data at a specified rate. The time required for the A/D conversion and the responsiveness of the overall system is collected. In the second lab, the students create the same data acquisition system that is interrupt-driven. In this case, the interrupt latency is measured and compared to the system timing of the polled I/O system. Creating the same functionality using two different approaches has proven to be a valuable technique in demonstrating important differences in performance and implementation. These two assignments also support many of the interfacing topics covered in the lecture portion of the course including general I/O configurations, writing interrupt-service routines, and decomposing bus-based communication into atomic bus transactions using master-slave relationships.

Another lab assignment that is used is that of creating a software simulation of a memory hierarchy. For this assignment, there is no direct connection to the VMEbus hardware, although students are encouraged to write their simulations using the lab stations to promote further familiarity with those systems. For this assignment, students are required to develop a simulation of a memory hierarchy configured according to user input. Once configured, the simulations must be able to accurately track memory performance given a set of memory references. Considering that many embedded applications have

predictable workloads, memory performance prediction and configuration is a necessary component of embedded systems development.

The final laboratory assignment involves real-time scheduling. Like the previous lab, the students are asked to develop a software simulation of a real-time scheduler configured according to user input. Possible configuration options include preemption or non-preemption, static or dynamic priority assignment, periodic or aperiodic task execution, independent tasks or tasks having precedence constraints. This assignment incorporates many concepts discussed in the IEEE/ACM model curriculum and included as part of the lecture material. For example, real-time operating system issues are addressed, as well as different priority assignment algorithms such as rate-monotonic and earliest-deadline-first. Scheduling processes also ties back into the concurrent process model of computation mentioned earlier as a technique used to describe system behavior. Students can now see the effects of different functional decompositions and different granularities of decomposition.

4. Future changes to ECE 486/487

After the first complete offering of this course with its associated laboratory assignments, it is evident that several adjustments must be made. First, a complete co-design laboratory assignment must be produced to complement the lecture material on this subject. Co-design is a rather abstract topic for students to understand especially if they have little to no design experience. The problems encountered up to this point with introducing such an assignment include finding a suitable system with the scope appropriate for a 1-2 week assignment, a system that will provide obvious and limited design choices after using trade-off analysis, and conquering the learning curve associated with design environments using co-design.

A second addition to the course includes expanding the software simulation assignments to incorporate the VMEbus systems. Adding a real-time operating system to one SBC will make it easy to incorporate the VMEbus systems into the scheduling assignments. Also, the VMEbus SBCs have cache memories and configurable caching options including the ability to turn caching off to support hard, real-time applications. With limited effort, it should be straightforward to incorporate the VMEbus systems into the memory simulator assignments.

Finally, additional lab assignments must be introduced to complement other lecture topics such as multiprocessing. As embedded systems continue to increase in complexity, multiprocessing is becoming a necessary topic as opposed to an "advanced" topic and must be incorporated into the class. The VMEbus systems readily support multiprocessing and this must become a fundamental part of the course.

In addition to adding laboratory assignments to the course, the course lecture and lab materials must be generalized in such a way as to make them available for use by others. The generalized versions of the materials should incorporate feedback generated from student assessment of the current materials. Assessment strategies are currently being defined.

5. Conclusions

The University of Alabama has reformed its Computer Engineering curriculum in order to incorporate an embedded systems theme throughout its core course sequence. One large component of these changes involves the introduction of a senior-level lecture/laboratory combination course concentrating on embedded systems. This course is integrated into the core course sequence and its lecture topics are derived from the IEEE/ACM model computer engineering curriculum. The laboratory assignments are designed to complement the lecture topics, and they also incorporate many of the topics, both core topics and elective topics, mentioned in the model curriculum. The laboratory assignments make use of a system architecture designed around the VMEbus. The VMEbus is shown to provide a powerful, flexible platform from which to teach many of the concepts in the model curriculum.

6. References

[1] Gannod, G. C., Golshani, F., Huey, B., Lee, Y. H., Panchanathan, S., and Pheanis, D., "A Consortium-based Model for the Development of a Concentration Track in Embedded Systems", 2002 Proceedings of the American Society for Engineering Education Annual Conference and Exposition, session 1532.

[2] Wolf, W., "Rethinking embedded microprocessor education", In Proceedings of the 2001 American Society for Engineering Education Annual Conference and Exposition, Albuquerque, NM, 2001.

[3] Wolf, W., Madsen, J., "Embedded systems education for the future", In Proceedings of the IEEE, 88(1), pp. 23 . 30, January 2000.

[4] Stapleton, W. A., Ricks, K. G., Jackson, D. J., "Implementation of an Embedded Systems Curriculum" 20th International Conference on Computers and Their Applications (CATA'04), New Orleans, Louisiana: ISCA, pp. 302-307 (March 2005).

[5] Turley, J., "The Two Percent Solution," Embedded Systems Programming, December 2002, www.embedded.com/story/OEG20021217S0039.

[6] Ganssle, J., “A Call for a New Curriculum,” Embedded.Com, May 2002, www.embedded.com/story/OEG20020530S0075.

[7] “From Analysis to Action: Undergraduate Education in Science, Mathematics, Engineering and Technology”, National Research Council, National Academy Press, Washington, DC, 1996, http://www.nap.edu/catalog/9128.html.

[8] The IEEE Computer Society/ACM, Computing Curricula, www.computer.org/education/cc2001/.

[9] Hennessy, J., Patterson, D., *Computer Architecture: A Quantitative Approach*, 3rd Edition, Morgan Kaufmann, 2003.

[10] Georgia Institute of Technology, School of Electrical and Computer Engineering, http://users.ece.gatech.edu/~hamblen/4006/projects/nios_robot/ECE4006_page.html.

[11] IEEE Computer Society, Computer.org, Computer Society International Design Competition 2003, http://computer.org/CSIDC/.

[12] *IEEE Standard for A Versatile Backplane Bus: VMEbus*, ANSI/IEEEANSI/IEEE Std 1014-1987, 1987.

[13] Lewis, Daniel W., Fundamentals of Embedded Software, Prentice Hall, Upper Saddle river, New Jersey, 2002.