2006-273: LIONS AND TIGERS AND TESTING...OH MY!

Steven Barrett, University of Wyoming

Steven F. Barrett received the BS Electronic Engineering Technology from the University of Nebraska at Omaha in 1979, the M.E.E.E. from the University of Idaho at Moscow in 1986, and the Ph.D. from The University of Texas at Austin in 1993. He was formally an active duty faculty member with the United States Air Force Academy, Colorado and is now an Associate Professor of Electrical and Computer Engineering, University of Wyoming. He is a member of IEEE (senior) and Tau Beta Pi (chief faculty advisor). His research interests include digital and analog image processing, computer-assisted laser surgery, and embedded controller systems. He is a registered Professional Engineer in Wyoming and Colorado. He co-wrote with Dr. Daniel Pack "68HC12 Microprocessor: Theory and Application," Prentice-Hall, 2002; "Embedded Systems Design and Applications with the 68HC12 and HS12," Prentice-Hall, 2005; and "Microcontroller Fundamentals for Engineers and Scientists," Morgan-Claypool Publishers, 2006. In 2004, Barrett was named "Wyoming Professor of the Year" by the Carnegie Foundation for the Advancement of Teaching. Email: steveb@uwyo.edu

Daniel Pack, U.S. Air Force Academy

Daniel J. Pack is a Professor in the Department of Electrical Engineering at the United States Air Force Academy, CO. He received the Bachelor of Science degree in Electrical Engineering in 1988, the Master of Science degree in Engineering Sciences in 1990, and the Ph.D. degree in Electrical Engineering in 1995 from Arizona State University, Harvard University, and Purdue University, respectively. He was a visiting scholar at Massachusetts Institute of Technology-Lincoln Laboratory. He co-authored two textbooks on microcontrollers and embedded systems and authored over 70 journal and conference papers. He is a member of Eta Kappa Nu, Tau Beta Pi (faculty advisor), IEEE (senior), and ASEE. He is a registered Professional Engineer in Colorado. In 2005, Pack was named "Colorado Professor of the Year" by the Carnegie Foundation for the Advancement of Teaching. His research interests include cooperative UAVs, intelligent control, automatic target recognition, and robotics. Email: daniel.pack@usafa.edu

"Lions and Tigers and Testing…Oh My!"

Abstract

The proper testing of a digital hardware and software design is often considered a dry and boring task for instructors to teach students. Anecdotally, we found students also share this perception of this important concept. A design, however, is only as good as the test plan that validates and supports it. We realize that entire textbooks and courses have been devoted to this topic, but, often, an engineering program does not have room for a standalone course on this topic. In our institutions, we elected to emphasize and allow students to practice some of the basic tenets and proper procedures of testing and documentation in several senior and graduate level design, microcontroller and hardware descriptive language courses. In this paper we will briefly review the basic tenets of testing and documentation and present some innovative methods of extracting test data from a hardware/software based project often found in a digital controller based system. We discuss how these tenets and techniques were adopted in several senior level courses and the overall results.

Overview

In the classic movie "The Wizard of Oz" Dorothy, the Tin Woodman, the Cowardly Lion, and the Scarecrow are making their way through a dark, dangerous forest. Around every turn they are worried about what they might encounter. There could be "Lions and Tigers and Bears…oh my!" The proper testing and documentation of a digital based system is also fraught with a variety of "dangers." Frequently the subject of project testing, test plans, and documentation is often treated as a dry and boring task in academia. A tedious and monotonous task of extracting system data from a complex digital design such as an embedded controller has contributed to this view. However, we all know that it is one of the most important concepts that must be taught in the engineering discipline.

Our accreditation body, Accreditation Board for Engineering and Technology, Incorporation, Engineering Accreditation Commission (ABET/EAC), recognizes the importance of these concepts. In their "Criteria for Accrediting Engineering Programs,"

under the Curricular Objectives section, we find "an understanding of the engineer's responsibility to protect both occupational and public health and safety." According to ABET/EAC, the definition of design includes testing and evaluation of design. In addition, the importance of instructing design safety is also mentioned in several areas of the criteria [1].

As future engineers, our students must recognize the tremendous responsibility they have designing devices, products, and projects that will interact and be used by the public at large, in some cases dealing directly with human lives. Often testing is relegated to a requirement that must be accomplished as part of a laboratory assignment or for a software project. A student, who will remain anonymous, provided a seemingly typical feeling concerning testing and test plans, "oh yeah, I have to do a test plan for this project…I'll throw something together to satisfy the assignment." Students frequently view the test plan as a separate assignment they must complete without considering the system they are designing. We believe this is a common feeling among engineering students (and possibly some faculty). Closely related to the concept of testing is documentation. Often documentation such as structure charts, Unified Modeling Language (UML) activity diagrams, and laboratory notebooks are completed when a project is nearing completion rather than used as powerful design tools at the start of the project.

To remedy such problems, we decided to emphasize the basic tenets and practice of testing and documentation in two senior elective courses in digital design and in a senior design capstone course. All students within our discipline take one (or both) or more of these courses. Our goal is to seamlessly integrate the concepts of testing and documentation into the fabric of these courses. It was done so that students would view testing and documentation as a natural portion of any design activity.

In this paper we provide background information on the basic tenets of testing and documentation along with method to extract test data from complex digital systems. We then provide several case studies where these techniques were integrated into a senior level Verilog hardware descriptive language course, a senior/graduate level embedded systems design course, and a senior design capstone course.

Background

Testing. System testing is an integral step in the design process. A typical design flow is illustrated in Figure 1. As shown in the figure, system testing usually occurs late in system development. At its most fundamental definition, testing is simply verifying that a system meets its intended requirements and specifications. However, if the system will be operated or interact with humans (as most of them do), testing must ensure a system:

- will operate correctly when it is operated properly,
- will not operate adversely when incorrectly operated, and
- has the ability to recover from and operate safely in the presence of faults.

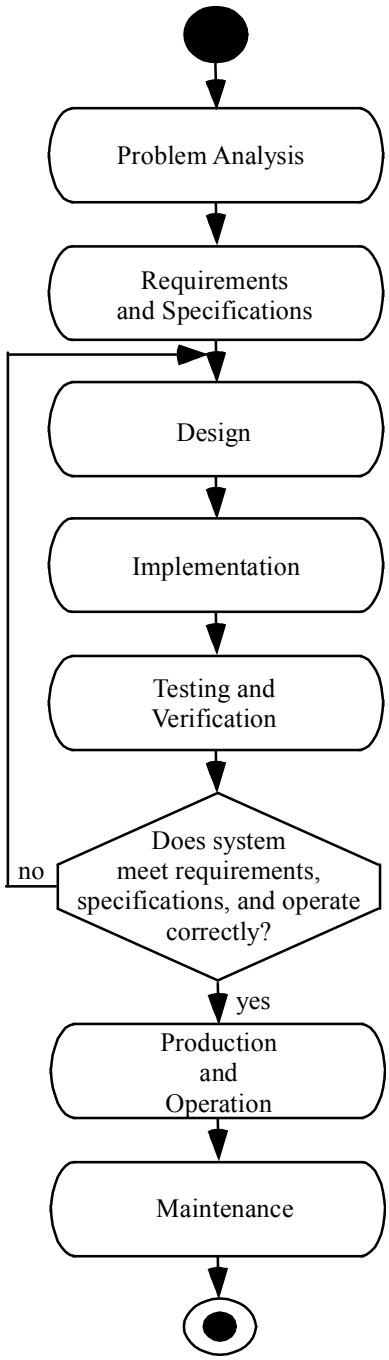

Figure 1. The design process [adapted from 2].

This definition implies exhaustive system testing. This concept may seem a bit overwhelming at first to both educators and students, but there are a few basic tenets of testing that when followed will go a long ways toward properly verifying the operation of a system.

For completeness, we will briefly cover these basic tenets of testing and also some techniques to test a complex embedded system which has significant hardware, software, and interface components. It must be emphasized that this is the barest of essential information that can be readily inserted into a course. For additional tutorial information on this topic, please see the references [2, 3].

Test Plan. The most fundamental component of system testing is the test plan. The test plan rigorously tests a system under a variety of conditions to insure it meets specified requirements and specifications and that it operates correctly under anticipated operating conditions. A test plan consists of a battery of system tests with expected and actual results documented.

Testing Approach. To thoroughly test a system and to detect system malfunctions as early as possible, an incremental approach is usually employed. A top down, bottom up, or hybrid approach may be used. In a top down testing approach, the overall framework of the system is first implemented and tested. Lower level system functions are deferred until the overall system framework has been tested and verified. This approach is particularly useful for a menu driven system. In this type of system a user will select different functions for the system to execute. The overall system framework (the menu or user interface) is first implemented and tested. Once this portion of the system is operating correctly, the lower level system modules could be added. For example, if we were designing a controller for a gasoline pump at a service station. We could design, implement, and test the pump's user interface first before implementing the actual lower level functions to operate the pumps, select the appropriate grade of gasoline, etc.

In a bottom up testing approach, lower level functions are first tested. These lower level functions are then integrated with other system lower level functions and an overall system is integrated up from these low level components. For example, if we were developing a robot control system, we would want to implement and test each robot subsystem first (vision, sensing, movement, etc.) before integrating them together. Many system designs lend themselves to a hybrid testing approach. In this case, top down and bottom up testing techniques are employed simultaneously.

We have been very careful in our description not to indicate whether it was hardware or software undergoing testing. In an embedded control system, implemented with an HDL or a microcontroller based system, both hardware and software are extensively used to implement the system. The testing techniques described are equally applicable to both.

Testing Techniques. Often a low cost controller (HDL or microcontroller) is used to control a complex, expensive hardware system. For example, if we were designing a controller for an industrial gate control application, we would want to ensure the controller's algorithm was functioning properly before integrating the controller with the expensive components. On the other hand, how can we test the control algorithm without connecting it to the hardware it will control? To solve this dilemma, a low cost system simulator may be constructed that provides simulated controller inputs. In response to these inputs, the system will generate appropriate control signals that may be measured

and analyzed with external indicators and test equipment to verify proper system operation. Figure 2 illustrates a generic, low cost hardware simulator. Debounced tact switches are used to simulate user input. Also, external analog voltages from trim potentiometers, tact switches and DIP switches input may be used to simulate external hardware response. System output is provided to light emitting diode (LED) indicators. In this particular example, the pulse width modulated signals generated at PD4 and PD5 by the controller may be observed. Both signals (PD4 and PD5) are fed into an operational difference amplifier so they might be simultaneously viewed on a single recording channel for testing purposes. By employing a low cost simulator a control algorithm can be completely and exhaustively tested prior to emplacement in an expensive circuit.

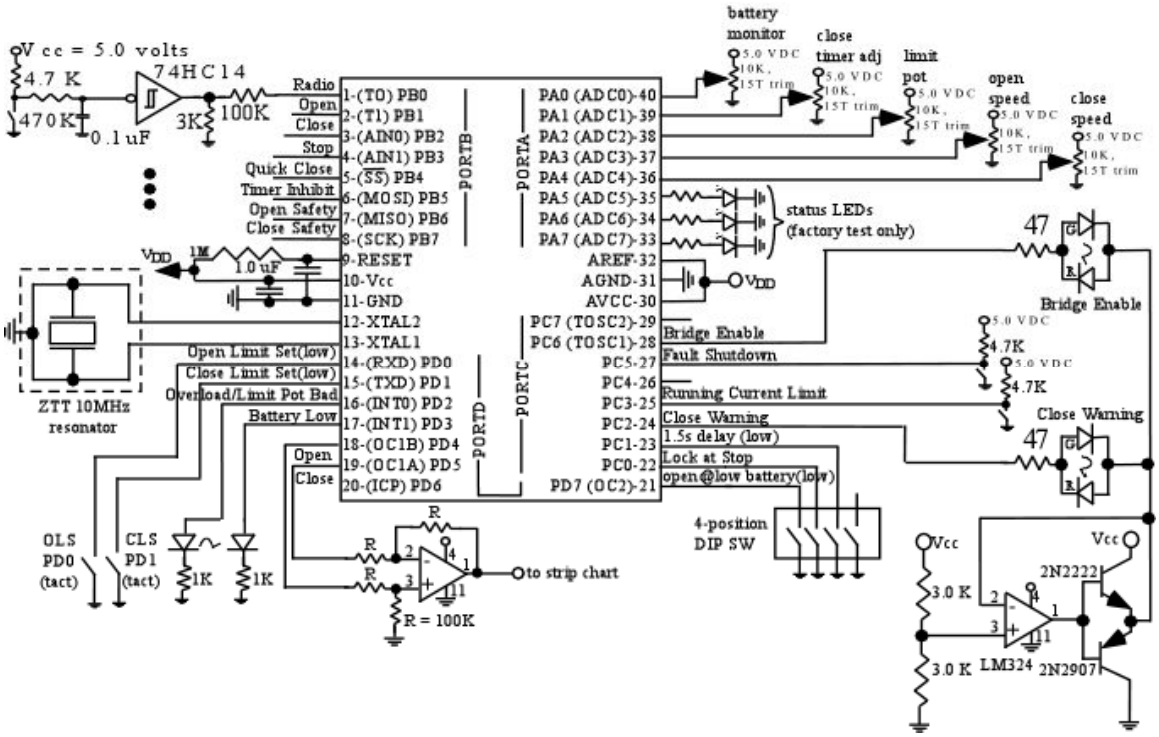

Figure 2. Low cost hardware simulator. The simulator consists of low cost tact and DIP switches, potentiometers, and LEDs to simulate expensive system components.

Testing Tools. There are a number of readily available test tools available to test a complex digital system. A brief definition of each is provided:

- Simulators/Testbench – There are a number of very good software integrated tool environments that allow for the design, testing, and synthesis of an HDL based controller. This allows a control algorithm to be fully tested before interfacing to external hardware.

- Emulators – An emulator, as its name implies, is a software tool that imitates the operation of a microcontroller. The emulator is hosted within a personal computer. An interface adaptor with the same profile of the embedded controller is placed in the system in place of the microcontroller. The emulator through the socket adaptor processes inputs and issues outputs to the target system.
- Oscilloscope – An oscilloscope is a good tool to measure the analog output of a microcontroller system. Oscilloscopes are commonly available that may display four analog outputs simultaneously.
- Logic analyzer – A logic analyzer is an effective tool for simultaneously measuring multiple channels (commonly 30+ channels) of digital data.

An embedded system designer commonly uses these tools to exhaustively test a system. However, in most control systems a mixture of analog and digital signals must be observed and their relationship to one another. We also need to extract the test data for a mixed mode system. Furthermore, often there is a severe limit on the number of microcontroller output pins available to output test points in the control algorithm. How do you do this?

In these cases, the simple "printf" statement may be used to print out test data to a host PC or possibly a liquid crystal display (LCD). These statements may be effectively used to print out code and data status. In very complex systems, they may be used to provide a record path through the executing code much like Hansel and Gretel marked their path with white pebbles. This allows the system designer to observe the flow of the algorithm during testing. The "printf" statement may require too much time to execute in reference to the code being tested and thus incorrectly influence test results. Furthermore, certain microcontroller products do not have the capability to "printf" back to the host computer. An LCD may be used for the "printf" instead but these are inherently slow and would suffer the same disadvantage of the "printf" statement.

To coax out tough internal status signals to external test equipment some variation of the circuit shown in Figure 3 may be employed. This circuit at a minimum requires only two microcontroller pins. One pin is used to shift serial data out of the microcontroller while the other required pin is a single bit control signal to latch the data. With this circuit internal data or progress points may be serially shifted out of the controller and into a 74HC164 serial-to-parallel converted. Once the data is finished shifting out, the latch is enabled (74HC573) to hold the data constant. If this circuit looks familiar, it is because it's the basic configuration of a serial input analog-to-digital converter.

The digital data is converted to a unique analog voltage for the digital pattern exported from the microcontroller. The analog voltage may then be displayed on an oscilloscope or strip chart recorder for slower algorithms typically found in industrial control applications. A unique analog voltage is associated with different points in the control algorithm. These analog data points provide a record path through the code.

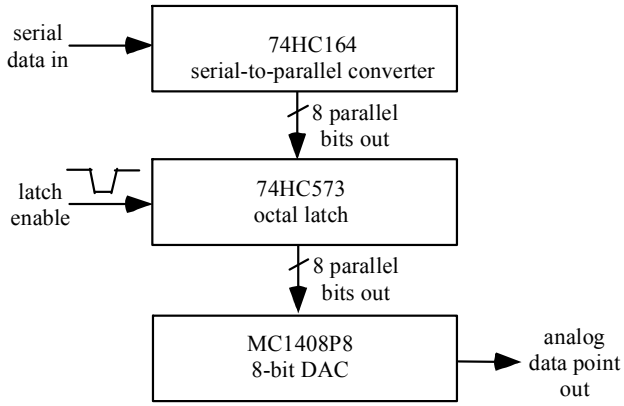

Figure 3. Auxiliary hardware to extract test data from a microcontroller.

Documentation. To thoroughly document the design, operation, and testing of a system; a number of documentation tools should be employed. Often these documentation methods are viewed as additional paperwork that must be completed after "the real" assignment, the code, is complete. This could not be further from the truth. Documentation tools should be used as powerful design and implementation tools to ease the transition of a system from a conceptual idea to a finished product. Here is a brief definition of each tool.

- Structure charts – a structure chart is the graphical tool used to compartmentalize a large complex project into hierarchical, definable related functions and subsystem black boxes.
- Pseudo code – pseudo code uses a "language like" description to define the operation of the black box. It provides an intermediate step between a word description and actual coding to conceptualize subsystem operation and interfacing.
- UML Activity Diagrams – a UML activity diagram is simply a UML compliant flow chart. These are used to describe the flow of an algorithm and its functions [4, 5, 6].
- Laboratory notebooks – laboratory notebooks are used to document the daily progress of a project. A well maintained laboratory notebook should document the who, what, when, where, why, and how of a design. Its entry should be properly documented in indelible ink with a signature, date, and witness [7].
- Test plan – a test plan documents the test results, insuring that a system meets its requirements, specifications, and intended operation.

Methods

The testing and documentation concepts discussed in the previous section were introduced into three different senior/graduate level courses to improve the design and testing skills of our electrical and computer engineering students. In this section we provide a brief description of each course and how the concepts were introduced.

EE4490 Hardware Descriptive Language. This is a senior level course in designing complex digital systems using CAD tools which target CPLD and FPGA devices. Prior

to implementation of the emphasis on testing and documentation, the course is used to instruct students complex hardware systems design knowledge. The student designs were scored against an instructor provided testing benchmark. Although, this was a good method of emphasizing sound design practices, students did not have the opportunity to develop and use their own test benches. To remedy this situation, (1) students were provided several lectures on structured design techniques and testing, (2) students were required to develop their own test bench to verify the proper operation of their hardware systems for all design assignments, (3) students were assigned to conceptu alize, set requirements, design, implement, and test a complex HDL based design, and (4) students were required to document their final project design with a written report and a detailed test plan.

All projects were required to be stand-alone. Projects ranged from commercial product controllers (coffee pot, camera, clocks, microwave oven, and elevators), to computer subsystems, to combination locks, and VGA controllers.

EE4590/EE5590 Real Time Embedded Systems. This is a senior/graduate level course in microcontroller systems design. This course emphasizes a systems approach to real time embedded systems. Students are expected to apply methodical system design practices to designing and implementing a microprocessor-based real time embedded system. Students employ a robot-based educational platform to learn the intricacies of real time embedded systems, distributed processing, and fuzzy logic. Students also learn processor input/output interfacing techniques Students use state-of-the-art design and troubleshooting tools.

In the course the students are required to complete a series of design exercises to equip a robot platform with the ability to autonomously navigate through a maze. To provide the necessary design skills for the course, students were provided formal instruction in structured design, testing, and documentation techniques.

To exercise these skills, student teams were required to design various subsystems for the robot platform including a vision system, a drive train system, and various operating systems to link the vision to the drive system. Operating systems based on polling techniques, interrupt driven systems, hybrid systems (polling with interrupts), and fuzzy based concepts were designed, tested an implemented. To emphasize structured design techniques, students were required to develop UML activity diagrams and structure charts for all laboratory assignments. In all cases students were required to document their design and results in their laboratory notebooks using hardcopy traces from appropriate test equipment. The robot platform and maze are provided in Figure 4.

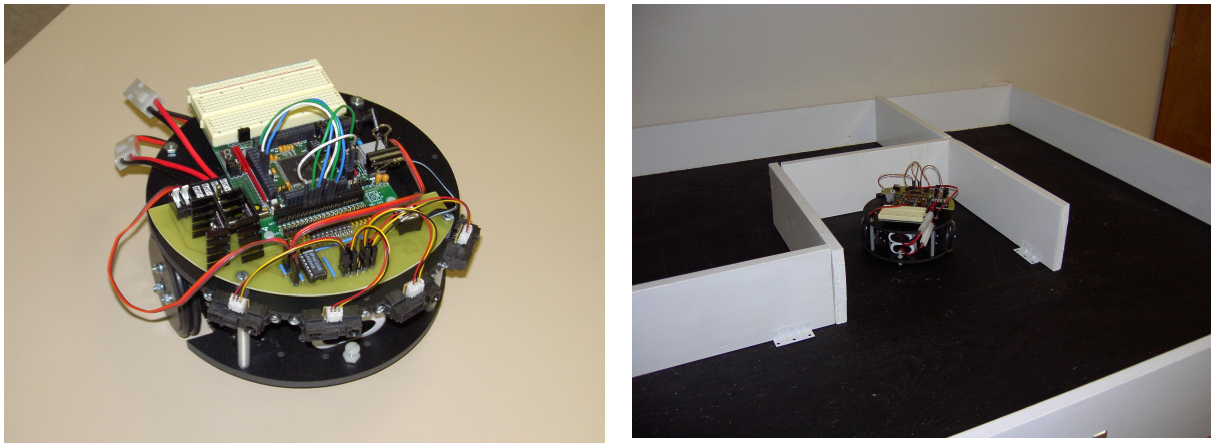

Figure 4. (left) Robot platform and (right) testing maze.

To interject competition into the course, each student team (17 total) competed against one another in the final laboratory exercise. Teams competed against one another to have their robot navigate through the maze as quickly as possible with time penalties for each wall collision. The winning teams received recognition for their design (bags of M&M's).

EE463/EE464 Senior Capstone Design. The year-long course is designed to provide students opportunities to design, implement, test, and evaluate a complex interdisciplinary projects. Throughout the process of completing a project, students are required to make six formal briefings accompanied by formal reports. Students are given descriptions of projects with a list of incomplete requirements for each project at the start of the course. Students are required to maintain and log their activities on lab notebooks throughout the semester. While the first semester is dedicated to complete detailed software and hardware designs of a project, students implement, test, and evaluate the project during the second semester.

Students turn in a test and integration plan as a part of the last formal briefing of the first semester. In the plan, we require that the document contains a detailed test plan which includes subsystem test plans and how to integrate subsystems together. At each level of progression, as students integrate subsystems together, we emphasize the importance of thorough testing and test planning to students.

In the second semester of the course we make the test report (test and integration plan) as a part of document turn-ins throughout the semester. The students turn in their first test and integration plan reports approximately one third of the way through the course, receive feedback from their faculty mentors, revise and resubmit their final test/integration plan reports approximately two-thirds through the course. As for the grades, 10% of their final grade for the second semester course (EE464) is allocated for the draft of the test/integration plan and the final version is worth another 15%, making the total of 25% of a student's grade.

Case Studies. In all classes, course instruction was supplemented with considerable examples from real world practices. Both authors have designed a wide variety of complex control systems for high end audio systems, commercial entry control systems, and a host of robotic applications. The case studies were used to emphasize the safety aspects and testing of a project. Also, it emphasized the structured design concepts taught in class. It is a powerful message to demonstrate to your students that you use the same design, testing, and documentation techniques for design as you teach in class.

Results

For the two elective courses, to measure how well the concepts were internalized by the students involved (4 courses, approximately 120 students), students were tested on the concepts via traditional tests and final examinations. Students performed well on describing key components of the concepts and applying them in design exercises. Students also did a good job of properly documenting their work in laboratory notebooks. Students were allowed to use their laboratory notebooks during tests and examinations. This was purposely done to encourage them to employ sound documentation techniques and also to emphasize the use of the laboratory notebook as a tool.

The most exciting results were in the final design projects in the Verilog HDL course and the senior capstone course. Students were required to complete an open ended, team project of their choosing. They were required to demonstrate they mastery of design and testing concepts through the project. The senior design projects were mentored by a team of faculty members who rigorously monitored students' progress to meet the course requirements which include detailed testing plans and meticulous documentation. The informal feedback from the faculty members show that the frequent formal report deadlines combined with a clear presentation of the testing and documenting process at the start of the semester were effective. The prior project results were of the highest caliber and demonstrated complex design depth. We found the great differences between teams that followed the strict requirements of testing and documentation and the ones that did not. Some of the projects that were successful are a telemetry recording system, cooperative mobile robots, and a wireless network system. In retrospect the students were motivated to pursue a project of their own choosing.

Discussion and Summary

Due to the success of these efforts in the two elective courses, the robot design project will be included in the EE4490 Hardware Descriptive Language course. Work is ongoing to develop a series of laboratory exercises that will allow students to develop a robot control system using HDL design techniques. Since many students take both courses, this will give them the opportunity to compare and contrast design procedures employing HDL versus a microcontroller based design.

Overall, student critiques were quite favorable for the courses that require testing and documentation. We would like to believe that all students understood the benefit of learning and applying the concepts of testing and documentation. Some student comments led us to draw this conclusion:

> "Great course, parts of this class should be taught in senior design."
>
> "Introduced a lot of good real world examples."
>
> "Good case studies."

On the other hand, some comments made it clear that we have additional work to do:

> "Assignments seemed like busy work."
>
> "Sometimes the homework seems like busy work."

Conclusions

We are encouraged by the overall results of our initial work to seamlessly integrate the concepts of testing and documentation in these courses. However, we feel there is additional work to do. We look forward to this challenge. The most positive feedback received was from students returning from a successful job interview and those who worked in project teams after graduation. The one with a job interview indicated that he was heavily questioned on the concepts emphasized in the course. He felt he received a job offer because of his demonstrated experience in this area. Those who visit after graduation feel that the course materials from these courses were most beneficial and relevant to their past and current jobs.

All material developed for these courses is available for your use. Feel free to contact us at steveb@uwyo.edu to obtain this material.

Acknowledgments

We would like to gratefully acknowledge the gracious support of the Electrical and Computer Engineering Department at the University of Wyoming for providing the seed funds to develop and build the educational robots and maze used in the course work

References

1. "Criteria for Accrediting Engineering Programs," ABET Engineering Accreditation Commission, December 19, 2005 (available at www.abet.org).
2. Barrett, Steven F. and Daniel J. Pack. Embedded Systems Design and Applications with the 68HC12 and HCS12. Upper Saddle River: Prentice Hall, 2005.
3. Abramovici, Miron, Melvin A. , Arthur D. Friedman. Digital Systems Testing and Testable Design. Hoboken: Wiley, 1994.
4. McCormack, John B., Robert K. Morrow, Harold F. Bare, Robert J. Burns, and James L. Rasmussen. The Complementary Roles of Laboratory Notebooks and Laboratory Reports. Proceedings: 1990 American Society for Engineering Educators Annual Conference, June 1990, Toronto, Canada, 1990.
5. Kobryn, Chris. ``UML 2001," Communications of the ACM, October 1999, Volume 42, Number 10, pages 29-37
6. Fowler, Martin and Kendall Scott. UML Distilled - A Brief Guide to the Standard Object Modeling Language. Boston: 2nd ed. Addison-Wesley, 2000.
7. Bruce Powel Douglass. Real-Time UML - Developing Efficient Objects for Embedded Systems. Boston: 2nd ed. Addison-Wesley, 2000.