Session 2793

A Practical Application Digital Systems Course For All Engineering Majors

Andrew J. Blauch, Andrew Sterian
Padnos School of Engineering
Grand Valley State University

Abstract

This paper discusses the development of the Introduction to Digital Systems course at Grand Valley State University. As this course is a prerequisite for all engineering majors prior to secondary admission, the course focuses on the practical application of digital systems to solve engineering problems. During the laboratory portion of the course, students design, build, and test various types of digital systems. A unique feature of the course is its integration of digital system fundamentals, C programming, and microcontroller interfacing. Logically and sequentially thinking in both hardware and software are enforced throughout this course. For the Mechanical and Manufacturing disciplines, this provides a strong foundation for the types of digital system applications that will be encountered in the upper level courses and senior projects. For the Electrical and Computer disciplines, the depth of content is developed in subsequent courses. The rest of this paper expounds upon the motivation behind the course, course topics, lab activities, and integration of the course into the ECE and ME/MFG curriculum.

Introduction

At Grand Valley State University, all pre-engineering majors are required to take a series of fundamentals in engineering courses before being admitted into the engineering program. The Introduction to Digital Systems course (EGR226) is one of these courses. As such, the course must be relevant to all fields of engineering taught at GVSU, including computer, electrical, mechanical, and manufacturing. Therefore, the course content has been developed to provide practical applications of digital systems that may be encountered by all engineering disciplines.

Over the past several decades, digital systems have become incorporated into many different types of products. Most modern industrial machinery and many commercial products are controlled by some type of microprocessor-based system. Many introductory digital courses now contain some type of controller-based application in their laboratory activities[1,2]. Because of the proliferation of microprocessor-based systems, it is important that all engineering graduates understand how to program and interface with these devices. To effectively do so requires a basic understanding of how these complex digital systems operate. To address this need, the content of this course has been developed around a unique set of topics. The three main topics covered in the course are digital system fundamentals, structured programming, and microcontroller interfacing. Digital system fundamentals is a standard component of introductory digital systems courses[3,4] and underpins the module on microcontrollers. The structured

programming instruction is narrowly focused to support the students' abilities to write microcontroller interface programs.

In addition to the practical skills developed in this course, the basic concepts are used to support upper-division courses in all engineering emphases. The junior-level Dynamic Systems course (EGR345) for mechanical and manufacturing engineering students depends upon the ability to write structured programs for implementing non-linear numerical algorithms. Senior-level courses in Manufacturing Controls (EGR450) and Integrated Manufacturing Systems (EGR474) available to mechanical and manufacturing engineering students depend upon digital system concepts such as Boolean logic and state transition diagrams, as well as experience with hardware interfacing. In the Electrical and Computer engineering emphases, of course, the dependence upon this introductory course is strong and all of the concepts are heavily used in various upper-division courses.

The prerequisite for EGR226 is only an introductory computer-programming course, in which Java is the current language of instruction with the emphasis on literacy of the object-oriented paradigm rather than programming skill. In addition, transfer equivalencies for this prerequisite course mean that students may arrive with varying knowledge of C++ instead of Java. We must, therefore, provide some instruction in C programming in EGR226 to ensure that all students have sufficient skills in structured programming.

Course Organization

The course begins with the fundamentals of digital systems and progresses toward the development of a general-purpose computer. During this development, the basic building blocks of a microprocessor are covered, such as adders, registers, and counters. One of the key outcomes of the development of the microprocessor is the shift from hardware design to software design. The end of the course focuses on the programming of a microcontroller and the development of software to solve simple engineering applications.

The structured programming portion of the course emphasizes logically and sequentially thinking in the development of software. Instead of waiting until later in the course to introduce programming, topics are intertwined with the fundamentals of digital systems portion of the course. For example, after number systems and representations are covered, data types and their relationship with number representations are presented.

The laboratory portion of the course is divided into two types or exercises: introductory activities and projects. The introductory activities are intended to familiarize the students with the physical devices and systems discussed in lecture. The procedures are fairly straightforward and guide the student through the desired activities. On the other hand, the projects are fairly open ended. Each project is based on simple digital systems developed in lecture or previous laboratory activities. The projects involve expanding these systems to accomplish a more practical application.

The following sections discuss the three main topic areas. Included in the sections are the course objectives, student learning outcomes, and summary of the subtopics, laboratory activities, and projects for that particular topic area.

Digital System Fundamentals

Course Objective

- Students will learn how to analyze and design basic digital circuits.

Student Learning Outcomes

- Students will be able to analyze basic combinational logic and synchronous sequential circuits.
- Students will be able to build, test, and debug logic circuits.
- Students will be able to design basic combinational logic and synchronous sequential circuits.

The material in this part of the course involves the introduction of digital signals, Boolean logic and algebra, truth tables, logic gates, storage elements, sequential networks, and finite state machines. In addition, we introduce the students to basic electronic laboratory equipment such as breadboards, power supplies, integrated circuits, DMM's, etc.

We present the design and analysis of combinational circuits as schematics, truth tables, and logic equations, along with the strategies for converting between these representations. This course module culminates with a discussion of the full adder and its role in forming a multi-bit ripple carry adder. For sequential circuits, we begin with one-bit flip-flops and combine them to form registers and counters, ending with a two-bit synchronous up-counter. This latter design is also used to demonstrate the concept of a state transition diagram and the process of translating such a diagram to a sequential circuit.

The laboratory activities in this topic area include exploratory exercises in combinational logic and sequential logic, and projects involving the design and implementation of a full two-bit adder and synchronous, two-bit up-down counter.

Structured Programming

Course Objective

- Students will learn how to write structured programs.

Student Learning Outcomes

- Students will be capable of designing a structured program from a problem statement.
- Students will be capable of writing a structured program.
- Students will be capable of compiling, executing, testing, and debugging a program.

As we mentioned previously, instruction in C programming is not a separate module but is taught throughout the course in support of the primary material. For example, after presenting Boolean logic for digital signals, we discuss the bitwise operators in C and their relationship to Boolean logic. In this way, we present a large portion of the basic C language. However, since C programming is not the sole objective of the course, we do not reach some of the more advanced concepts of the language such as structures, multidimensional arrays, etc.

We also require an increasing level of discipline as the course progresses with respect to using good structure, proper commenting, modularity, and robustness. By the end of the course, we expect that all subroutines be properly and consistently commented, programs are developed as modular components with well-documented interfaces, and reasonable efforts are made to handle all program and user-input errors. The majority of instruction in these programming practices is by example and correction of homework exercises. We very briefly discuss the use of flowcharting and pseudocode for high-level program development.

The laboratory activities in C programming are used primarily as opportunities for students to develop their skills in a supervised environment, where errant ideas can be quickly corrected. The first programming lab is used as an introduction to C and our development environment. The other C programming laboratories are used for the implementation of small projects. We have used two project ideas for this purpose in different offerings of the course. The first is the two-player game of Nim played over a TCP/IP interface. A basic socket library is provided for the TCP/IP interface. The students must understand the library API and develop the C program for playing Nim as either the "client" or the "server". The second project involves the development of a numerical analysis package of an unknown function, provided as a library. The students must develop code to find minima and maxima, compute the definite integral over a user-defined range, find the roots of the function in a user-defined range, etc.

For the C programming activities, we use the command-line Borland BCC32 compiler, version 5.5. This compiler is freely available thus allowing students to work on their assignments in any location.

Microcontroller Interfacing

Course Objective

- Students will learn the general structure and operation of a microprocessor-based system.
- Students will learn how to develop software for a microcontroller-based system.

Student Learning Outcomes

- Students will be capable of identifying the basic components of a microprocessor-based system.
- Students will be capable of defining the basic terminology, and describing the general operation, of a microprocessor-based system.
- Students will be able to establish communication with, and monitor the operation of an embedded microcontroller.
- Students will be capable of writing microcontroller specific programs.

This final module in the course is perhaps the most practical. By the end of the course, we expect that students will be able to write a C program for an embedded microcontroller system. Our main vehicle for this module is the 68HC11 microcontroller embedded in the Axiom CME11-E9 EVBU development system. These systems are inexpensive, have 32K of RAM, a small breadboard development area, and thus are a perfect fit for the goals of the course. For programming, we use the free GCC compiler targeted for the 68HC11.

After presenting the basic components of a microprocessor/controller, we begin discussing the specific architecture and peripherals of the 68HC11. We use the output compare and input capture peripherals extensively to present the concept of real-time in sensing and control. For example, we discuss the generation of accurate pulse-width modulation waveforms with a given duty cycle, and the accurate estimation of frequency of a periodic digital waveform. We conclude this module with a discussion of the concepts and mechanics of interrupt-driven peripheral interfacing.

The first lab for this module introduces the students to the 68HC11 Axiom CME11-E9 development system. Subsequent laboratories include monitoring the state of parallel I/O, user interface to a seven-segment LED, and implementation of a frequency monitor using input captures. The final project requires the students to design and implement a mock controller for an industrial process. The set-point of the process is determined by a potentiometer, which is sensed using the 68HC11 analog-to-digital converter. The program must then control the output frequency of a square wave based upon the set-point. In various offerings of the course this output has either been used to drive an audio speaker or DC motor.

Sample Laboratory Activity

A sample of one of the laboratory activities is provided on the following page. This laboratory activity is conducted towards the end of the course and is intended to familiarize the students with the input capture feature of the 68HC11. The Axiom CME11-E9 EVBU development system is connected to a PC via a serial connection. In a previous laboratory activity the students learned how to communicate to the EVBU, compile and download programs, and transmit/receive data across the serial cable as part of their own programs. For this lab, as a practical application, the students are directed to use the input capture feature of the 68HC11 to implement a digital frequency monitor. As with most of the programming projects, the students are given a list of additional features that are to be added as time permits. At the end of lab the students are required to demonstrate their working programs.

EGR226 Laboratory Activity
68HC11 Input Captures

Objectives

- To control the input capture features of the 68HC11.
- To build a digital frequency monitor.

Introduction

The measurement of frequency involves counting how many events occur within a given time period. These events may be the zero-crossings of a sine wave, or the rising edges of a digital waveform. We can see that measuring frequency involves two basic activities: counting events and keeping time. Both of these activities are supported by the on-chip peripherals of the 68HC11. This support allows us to write code that performs accurate measurements with a minimum amount of effort.

Measuring the frequency of an oscillating waveform may be an end unto itself; this application is known as a frequency counter. In addition, there are several applications in which the frequency of an oscillating waveform is related to a different physical attribute, such as temperature or capacitance. Frequency can also be used to refer to how many components pass a conveyor mechanism in a given unit of time (sensed using an optical interrupter), thus providing a measure of the efficiency of an assembly line.

Materials

- EVBU kit, PC, CADET, Assorted wires

Procedure

As you work your way through this handout, be sure to record useful information (such as pin outs, schematics, etc) in your laboratory notebook. Remember, your laboratory notebook should be your primary reference – not the lab handouts. Any questions posed in this handout should be clearly indicated and answered in your laboratory notebook (i.e. copy the question in your notebook followed by your answer).

Periodic Input Signal

The function generator on the CADET unit will be used as the input signal to your frequency monitor. First, be sure the power is disconnected from the EVBU and turned off on the CADET unit. Connect the ground of the CADET to a GND pin of the EVBU. Connect the TTL output of the function generator on the CADET unit to the IC2 pin of the 68HC11. Do not connect the 68HC11 to any of the other outputs of the function generator. Just to be safe, turn the AMP dial all the way down. Set the frequency to approximately 10 kHz.

Main Program

Write a program that contains an initialization section and a polling loop. In the polling loop, just check for an input from the keyboard. If the letter ‘Q’ is pressed, exit the program. Compile, download, and execute your program. Make sure it works properly.

Frequency Monitor Program

Using the input capture I/O features of the 68HC11, devise a method for measuring the period of the input signal. Determine which events and pins you will use, how you will take the measurements, and in what order you will do it. Take some time and think about what you want to do and how you will do it (i.e. write pseudo-code or flowcharts) before you begin adding code. In the polling loop, your program should take one measurement of the period and display the time in counts. Compile, download, execute, and debug your program. Work in increments. Don’t try to add too much code at once. When it is working properly, demonstrate your program to your instructor.

As time permits, incorporate each of the following into your program.

- Display the period in microseconds. (Don’t use the printf( ) subroutine.)
- Display the frequency in Hertz. (Don’t use the printf( ) subroutine.)
- Determine the minimum and maximum frequencies that your program can measure.
- As best you can, explain the reasons for the minimum and maximum frequency limits of your program.
- Add a one second delay (using either the RTI or TO events) after each measurement.

Assessment

The main assessment mechanism for this course is its impact on the upper level courses. The upper level courses affected by this course can be separated into two groups: Electrical Engineering courses and Non-Electrical Engineering courses. A brief summary of how these two groups of courses have been affected is provided.

The Electrical Engineering course that follows directly from EGR226 is EGR326, Advanced Digital Systems. In this course, students are required to complete a project: the design and build of a complete microcontroller-based system. The experience of writing structured programs for a microcontroller system in EGR226 allows the EGR326 project experience to be much more mature than would otherwise be possible. For example, students routinely incorporate PS/2 keyboard interfaces, motor interfaces, LCD display interfaces, etc. in their EGR326 projects without any supporting instruction.

The combined effect of EGR226 and EGR326 supports senior-level courses, such as EGR423, Digital Signal Processing. In this course we have been able to introduce a project component that involves both DSP software and hardware interfacing. Without the experience of EGR226, followed by EGR326, the DSP course would be limited in its ability to provide practical design experience.

The Non-Electrical Engineering courses that are influenced in some part by the content of this course are Dynamic Systems and Control (EGR345), Manufacturing Controls (EGR450), and Integrated Manufacturing Systems (EGR474). The junior level course (EGR345) builds upon the structured programming aspect. Students are required to write numerical algorithms for solving non-linear systems of equations. In addition, students learn how to develop LabVIEW applications to collect and analyze experimental data. The structured programming techniques emphasized in EGR226 are essential to the successful and efficient completion of these assignments.

The two senior level courses (EGR450 and EGR474) deal with the programming of various industrial controllers and networks. In order to intelligently work with these types of system, it is imperative that the programmer be well versed in the structure and operation of microprocessor-based systems. By tying the topics of digital system fundamentals (Boolean logic, sequential logic), structured programming, and microcontroller interfacing together in one cord, the students come out of EGR226 with a strong foundation upon which to build these two courses.

Conclusion

The primary purpose of this course is to provide practical applications of digital systems to solve problems relevant to all engineering disciplines. Digital system fundamentals, structured programming, and microcontroller interfacing are all integrated into the course content. The laboratory activities introduce the students to the various concepts and systems discussed in lecture. The projects are used as a mechanism for the students to apply the concepts they have learned in a logical and sequential manner to solve practical engineering problems. The course

serves as a foundation for the upper level engineering courses by providing the basic understanding, programming skills, and interfacing ability necessary in order to work with microprocessor-based systems.

Bibliography

1. O. Fucik, B. Wilamowski, and M. McKenna, "Laboratory for the Introductory Digital Course," *Proceedings of the 2001 ASEE Annual Conference and Exposition,* Albuquerque, New Mexico.
2. G. Foster, "Laboratory Design Projects for a Freshman Digital Electronics Course," *Proceedings of the 2001 ASEE Annual Conference and Exposition*, Albuquerque, New Mexico.
3. D.J. Tylavsky, "An Introductory Digital-Logic Design Laboratory," *Proceedings of the 1999 ASEE Annual Conference and Exposition*, Charlotte, North Carolina.
4. B.S. Motlagh and A. Rahrooh, "The Fundamental Digital Circuits Laboratory at The University of Central Florida," *Proceedings of the 1999 ASEE Annual Conference and Exposition*, Charlotte, North Carolina.

ANDREW J. BLAUCH

Andrew J. Blauch is currently an Assistant Professor in the Padnos School of Engineering at Grand Valley State University. He received his B.S. in Electrical Engineering from Messiah College, M.S. in Electrical and Computer Engineering from Carnegie Mellon University, and Ph.D. in Electrical Engineering from the Pennsylvania State University. He has taught courses on digital systems, microprocessors, and controls.
<http://claymore.engineer.gvsu.edu/~blaucha>

ANDREW STERIAN

Andrew Sterian is currently an Assistant Professor in the Padnos School of Engineering at Grand Valley State University. He received his B.A.Sc. in Electrical Engineering from the University of Waterloo, Canada and the M.S.E. and Ph.D. in Electrical Engineering from the University of Michigan, Ann Arbor. He has taught courses in signal processing, digital systems, and microcontrollers.
<http://claymore.engineer.gvsu.edu/~steriana>