IN-CLASS DEMONSTRATIONS WITH A PORTABLE LABORATORY FOR TEACHING DSP TO COMPUTER ENGINEERING MAJORS

Andres Kwasinski
Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA.

ABSTRACT
Given its pervasive use, and being a clear example of a computing system integrating hardware and software, Digital Signal Processing (DSP) is an important discipline in Computer Engineering undergraduate curricula. This paper presents a DSP course that, in addition to traditional lectures and exercises, uses Matlab to provide a limited level of laboratory-type practice. Also, the use of demonstrations during lectures is presented. The presentations help the students relate the theoretical concepts presented in class with the technical area of expertise they are most comfortable with, that of hardware, software and their integration. The demonstrations are implemented using a portable laboratory with a DSP board and a small oscilloscope. Presented results show how the use of demonstrations has increased the overall rating for the course and how stimulating it is found.

Index Terms— DSP Education, Computer Engineering.

1. INTRODUCTION
Discussing teaching DSP to Computer Engineering majors implies, to an extent, understanding what Computer Engineering involves and identifying students’ areas of strength. In 2004, a joint IEEE/ACM task force created, with the support from the National Science Foundation, a study detailing curriculum guidelines for undergraduate degree programs in Computer Engineering [1]. The report defines Computer Engineering as “the discipline that embodies the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems and computer-controlled equipment”. Traditionally, Computer Engineering has evolved from the intersection of Electrical Engineering and Computer Science. The domain of Computer Engineering is that of designing, producing, testing and maintaining computing hardware, software, networks, applications and processes. Most importantly, this background and professional competencies translates into computer engineers having the expertise to address technical challenges in software, hardware and their interaction.

Given its nature of intimately linked hardware and software to process signals, DSP is a natural core discipline in Computer Engineering [1]. Furthermore, given its pervasive use in many modern technologies, Computer Engineers need to be intimately familiarized with the concepts learned in typical undergraduate-level DSP courses [2]. With a Department of Computer Engineering founded 35 years ago, the Rochester Institute of Technology (RIT) has a long standing commitment to the formation of Computer Engineers. The Computer Engineering program at RIT has recognized the importance of DSP education and includes a core course on DSP in its undergraduate curriculum. At the same time, the RIT has a renowned co-op program that requires students to engage in five quarter-long blocks of co-op work. The co-op program is highly valued within the RIT community for its important educational advantages (which are beyond this paper to discuss). At the same time, the co-op program introduces some particular considerations into the curriculum design. Undergraduate programs are designed with a length of five years to accommodate for the co-op blocks but still there is no opportunity within the busy curriculum schedule for the DSP course to include a full laboratory component as it is found in other DSP courses [3, 4]. At the same time, co-op considerations also influence when different courses are scheduled. This is because it is desirable that by the time students starts with their co-op experiences, they have already taken a breadth and depth of courses that allows them to have flexibility in the type of work they may choose as well as enough technical competency in those areas so that the co-op experience is of value for the students and the employers alike. As a result, the DSP course is typically taken by students in their fourth year of studies (co-op experiences start in the third year of studies).

At the time students at RIT take the DSP course, they have been one year or more removed from the courses in the math sequence. The time in-between finishing with the math sequence of courses and taking the DSP course is occupied with a variety of courses related with software and hardware competencies (e.g. Assembly Language, Applied Programming, Operating Systems, Hardware Description Language, Electronics, Computer Organization, etc.). As a result, many students taking the DSP course feel that they need to regain the familiarity with theoretical math, yet, they show an intimate understanding of software, hardware and their interworking in computing systems. This is both a challenge and an opportunity. Other challenges include striking the right balance between time limitations and depth of coverage of a topic, organizing the order in which the topics are covered so that time is used best and students find a natural flow of concepts, etc. More importantly, these settings translate into a high pace course for which a particular concern is that the students achieve a high level of understanding of all the topics.
covered. Because of this, it is important to be able to maintain a level of student’s motivation conducive to an effective and efficient learning experience. In this paper we describe the approach we have taken at RIT to teach DSP by creating an integrated learning experience. We will focus on describing the use of a portable laboratory that we use to do in-class demonstrations and partially compensates for the impossibility of having separate laboratory sessions. In addition, we present results from the students’ evaluations that show that the introduction of the demonstrations effectively addresses the students’ strengths and increase the overall rating for the DSP course and how stimulating students found it to be.

2. DSP AND THE COMPUTER ENGINEERING CURRICULUM

Leveraging the existence of a department of Computer Engineering, at RIT, the Computer Engineering curriculum has been exclusively designed for this major [5]. Digital Signal Processing is one of the core courses in this curriculum. Students take the DSP course following an applied programming course, which in turn has as prerequisite a course on linear algebra and a first course on differential equations. Other courses that the students have taken by this time include four courses on computer science, a software engineering course, assembly language, hardware description language, applied programming, computer organization and operating systems. All the engineering programs at RIT are organized based on a calendar with four quarter-long terms, with ten weeks of classroom instruction plus one week for final exams in each quarter. The following is the weekly schedule followed for the DSP course:

1) Concept of signal; complex number and complex exponential; introduction to Matlab.
2) Fourier Series.
3) Fourier Transform and its properties.
4) Continuous-time linear systems; convolution.
5) Sampling, aliasing and reconstruction.
6) Discrete-time linear systems; FIR filters.
7) Frequency response of discrete-time linear systems.
8) Discrete-time Fourier Transform and Z-transform.
9) Properties of Z-transform and response of discrete-time linear systems; IIR filters.
10) Poles and zeros; Discrete Fourier Transform.
11) Final exam.

Note that the CE curriculum does not include a course on Signal and Systems and, thus, the DSP course serves the broader purpose of teaching the traditional topics in an introductory DSP course (e.g. Z-transform, FIR and IIR digital filters, discrete-time Fourier transform, etc.) and also important topics traditionally taught in a Signal and Systems course (e.g. Fourier Series and Fourier Transform). Since the DSP course is not a prerequisite for other courses, students usually take this class in their fourth or fifth year of study. The concepts learned in the DSP course are later applied by several of the elective courses taken by the students, such as computer vision, image processing and digital control.

3. A MULTI-PRONGED APPROACH TO PRESENTING DSP CONCEPTS

Our main approach to presenting the different material for the DSP course has been based on traditional lectures, followed by homework exercises for reinforcements. Although in our curriculum, students take the DSP course in their fourth or fifth year of studies, we chose DSP First [6] as textbook due to its good match in terms of the assumed prerequisites and the covered topics.

A crowded curriculum does not allow for the DSP course to incorporate a full-fledged laboratory. Yet, this important component is addressed in two different ways. Firstly, the course incorporates two Matlab-based projects. Since it is assumed that students have no previous knowledge of Matlab, it is introduced early in the course. Approximately half of the first week is spent explaining the basics of Matlab. From this point on, Matlab is integrated into the course, first in homeworks and then as the target tool for two projects. The first project focuses on ideal sampling. In this project, students are asked to approximate with Matlab the behavior of an ideal sampling and reconstruction system and then experiment with different settings for sampling and reconstruction. The project also allows the students to gain more familiarity with Matlab in a medium size task. The second Matlab project focuses on modeling and understanding linear time-invariant discrete-time systems. The project starts with the "black box" problem, where the students are provided with three pre-compiled Matlab-callable functions. These functions represent a system with a single input and output. Because they are pre-compiled functions, the students have no way of reading the function code to know what it does. Then, the students need to use the concepts explained in class to decide on the different signals to inject the systems so as to be able to characterize and classify the systems in terms of linear/non-linear qualities and types of digital filters. Following this task, the project guides the students through different exercises intended to experience the duality of analysis in time or frequency and the interrelation between different systems. The project finishes by guiding the students through a simple design of a digital filter and asking them to implement the filter as a compiled C function, callable from Matlab. This task also presents the advantage of requiring that the students apply a number of the concepts taught during the lectures to complete the debugging process.

During last year we have incorporated a second component to our approach for providing contact with practical implementation issues. We have designed a portable laboratory to do demonstrations during lectures. Figure 1 shows the laboratory set up. Setting up the laboratory takes no more than five minutes. The central component of the laboratory is a Spectrum Digital TMS320C6713 DSP Starter Kit (DSK) [7], donated by Texas Instruments Inc. The kit consists of a Texas Instruments TMS320C6713 DSP [8] connected to a Texas Instruments TLV320AIC23 audio codec, which per-
forms analog-to-digital and digital-to-analog conversion to input and output signals to and from the DSP, respectively. The Texas Instruments TMS320C6713 is a single-core floating point DSP capable of delivering up to 1800 million instructions per second (MIPs) and 1350 million floating point operations per second (MFLOPS). Floating point implementation is useful in our case because it simplifies the implementation, as there is no need to deal with fixed-point logic issues. Alongside the actual hardware running the simulation, the demonstration equipment uses a USBInstruments DS1M12 “Stingray” portable Digital Oscilloscope that presents results on a PC (a Lenovo Ideapad S10-3t tablet netbook in Fig. 1) through an USB connection. The oscilloscope has two input channels and one output channel to generate simple signals. The oscilloscope has an input channel bandwidth of up to 250 KHz, which is adequate for any of the signals used in the demonstrations (with bandwidths in the range of audio signals). For those demonstrations that use the DSP (all but one, see below), we dedicate time to explaining the process of programming the DSP board, showing the source code for the demonstration and explaining the use of the programming environment (TI’s Code Composer Studio v.4.0, in our case) to run applications on the board. The laboratory is used to show four demonstrations as an integral part of the lecture. This allow us to enhance the lectures by making them more interactive, practical and stimulating for the students and at the same time, leverage on the strengths of Computer Engineering students, by discussing hardware and software issues. Because of this, and as the demonstrations consist of running applications on a DSP board, these activities can be thought as occupying a place in-between the traditional in-class demonstrations and a full fledged set of laboratory assignments. The four demonstrations that are presented are:

1) Using the Fourier Series synthesis equation to synthesize a periodic square-wave signal: For this demonstration, it is possible to generate a periodic signal by adding one, two, three or four harmonics. The number of harmonics that are combined at any time can be changed while the program runs by using a dip-switch bank on the DSK board. The demonstration is also used to discuss Gibbs oscillations.

This demonstration is presented during week 2 of the course.

2) Continuous-time systems impulse and frequency response: Presented during week 4 of the course, this demonstration is the only one that does not use the DSK board. The purpose of this demonstration is to illustrate and guide students through the process of deriving the frequency response, first, and the impulse response, second, for a linear time-invariant continuous-time system. The first step of this demonstration involves deriving the frequency response of an unknown system (a black box that, unknown to the students, contains a resistor-capacitor first order, low-pass filter). The system frequency response is derived by combining the responses to individual sinusoidal input signals. After this, it is explained how superposition can be used to obtain the response to any signal. These steps helps also put in perspective the reason for studying earlier the representation of signals as linear combination of sinusoidal signals. To conclude this demonstration, the convolution/multiplication property is used to derive the concept of a system impulse response.

3) Sampling theorem: This demonstration uses the DSK on-board analog-to-digital and digital-to-analog converters to exemplify the Shannon-Nyquist sampling theorem with practical devices. The system operates with a sampling rate of 8 KHz and the frequency of a sinusoidal input signal is changed to show cases with and without aliasing distortion. Each case is also explained on the board by deriving the spectrum of the different involved signals. This demonstration is presented during week 5 of the course.

4) Digital filters: This demonstration, presented during week 7 of the course, illustrates design and implementation of Finite Impulse Response (FIR) filters. Being this the last of the demonstrations, it goes into more details regarding design and implementation. The demonstration starts by illustrating how Matlab can be used to design a digital filter (an order twelve, low-pass FIR filter). The demonstration continues explaining the algorithm that implements the FIR filter and the architectural features in the DSP that allows highly efficient implementation of the algorithm. The demonstration of the filter includes measuring at different points, the frequency response of the filter and comparing with the predicted response obtained during the design stage. The demonstration finishes by showing a simple reverberation filter. This is an order-480 FIR filter that simulates the reverberation of sounds occurring in concert halls. The demonstration plays a sample of an audio song, which is processed through the reverberation filter and compared against the same sample without being processed. The students can perceive that the processed sample has a more rich sound, as can be heard in a concert hall.

4. EVALUATION RESULTS

As explained in previous sections, the introduction of a portable demonstration laboratory had the dual goal of illustrating practical issues to the students as close as possible to what laboratory classes allow, and also introducing a component to the
lecture presentation that increases student involvement and interest in the course. The rationale behind this approach is that the demonstrations address issues with hardware and software that are the biggest strength of computer engineering students.

To evaluate the level of success with the second goal, we compared students evaluations scores from the year before and after the introduction of the in-class demonstrations. These were the years 2008 and 2009, respectively. Figure 2 shows the overall rating for the DSP course. The Figure shows a clear and consistent improvement in the score after the introduction of the demonstrations. Because students exhibit similar experiences for the same quarter from year to year, it is also interesting to compare the scores for the same quarters. Here again the improvement is clear. We believe that the most important reason for this improvement is that with the introduction of the demonstrations, the students find an element they are familiar with (a computing device) illustrating concepts that are based on math theory they haven’t had much contact with for about a year. Figure 3 shows the scores of how stimulating the DSP course was for the students. The answer is in a scale from 1 to 3, with the score of 3 corresponding to the highest stimulating score. The Figure shows, that while before the introduction of the demonstrations there was always a percentage of students that answered with the lowest score, after the introduction of the in-class demonstrations no students answered with the lowest score.

5. CONCLUSIONS

We have discussed our design for a core course in DSP for a Computer Engineering curriculum, where program constraints do not allow for a full-fledged laboratory. The presented approach is still comprised of the traditional lecture plus homework component. In addition to this, a limited laboratory experience is introduced through two Matlab-based projects. Furthermore, lectures are complemented through in-class demonstrations using a portable laboratory built around a Texas Instruments TMS320C6713 DSK. The rationale to introduce the demonstrations is that they help the student relate how the theory from the lectures relate to practical implementations and also, by presenting applications of computing systems involving hardware and software, they played into the strengths of Computer Engineering students. We have shown evaluation results that show that the introduction of the demonstrations have made the DSP course more stimulating for the students and have increased their overall rating of the course.

6. REFERENCES