REAL-TIME DSP FOR SOPHOMORES

Kenneth H. Chiang *Brian L. Evans* *William T. Huang* *Ferenc Kovac*
Edward A. Lee *David G. Messerschmitt* *H. John Reekie* *Shankar S. Sastry* *

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720-1770
E-mail: *eecs20@hera.eecs.berkeley.edu*
WWW: http://www-inst.eecs.berkeley.edu/~ee20

ABSTRACT

We are developing a sophomore course to serve as a first course in electrical engineering. The course focuses on discrete-time systems. Its goal is to give students an intuitive understanding of concepts such as sinusoids, frequency domain, sampling, aliasing, and quantization. In the laboratory, students build simulations and real-time systems to test these ideas. By using a combination of high-level and DSP assembly languages, the students experiment with a variety of views into the representation, design, and implementation of systems. The students are exposed to a digital style of implementation based on programming both desktop and embedded processors.

1. INTRODUCTION

We are developing a new class, "Introduction to Real-Time Digital Systems," that combines signal processing and computer architectures in a laboratory setting to excite sophomores about communications, signal processing, and controls. The students experiment with sampled speech, music, and image signals to gain experience in analyzing, enhancing, and performing real-time processing on them. They learn about Fourier analysis at an intuitive level as the key technique to unlock the composition of a signal, but also experience its usefulness in practice. The students explore many different views of signals and systems as they build signal processing algorithms using high-level and assembly languages. They also gain practical experience developing algorithms in MATLAB [1] and embedded applications on Texas Instruments TMS320C50 boards [2, 3]. In demonstrations, students are also exposed to the visual block diagram programming environments SIMULINK [4] and Ptolemy [5].

The students for the course are sophomores majoring in Electrical Engineering and Computer Sciences (EECS). At Berkeley, the fields of electrical engineering, computer science, and computer engineering are taught in a single EECS Department. We have integrated these fields in the EECS courses so that they look seamless to the students. Thus, students are able to combine these fields in different ways, which is particularly appropriate in the context of modern technology. This new sophomore course offers students an exposure to a combination of communications and signal processing with computer science and engineering. The course comes at a time when the students are deciding on their areas of specialization from among electronics, systems, and computer science and engineering.

Regardless of their final area of specialization, all EECS students can benefit from the course. They gain an appreciation for real-time discrete systems and a digital style of implementation. Their understanding of discrete-time systems complements the analog and digital circuit design and desktop software programming they are learning in their other lower-division classes. Because we introduce systems by way of applications, interesting concepts, and digital computing, we hope to motivate students to study communications, signal processing, and controls. For those students who choose the systems area as their specialization, their practical understanding of concepts such as the frequency domain, sampling, aliasing, and quantization will give them better motivation to study the theory in later systems classes, because they have a greater appreciation of the application of the theory beforehand.

The initial offering of this real-time DSP course is during the Spring 1996 semester. This two-credit course runs for sixteen weeks, as shown in Tables 1 and 2 on the next page. The course consists of one hour of lecture, one hour of discussion, and four hours of laboratory work each week. Throughout the course, concepts and applications are interwoven. Each lecture demonstrates applications to illustrate concepts being presented. In each laboratory, students further their understanding by developing signal processing systems. The students spend 3-5 weeks on each of the following application areas: computer music and digital audio, speech, digital communications, and image processing. Within each application area, students build simulations and real-time systems. In developing the laboratories for this course, we found several books on MATLAB [6, 7, 8] and the family of Texas Instruments fixed-point DSP processors [3, 9, 10] to be very helpful.

2. LOGISTICS

In the initial offering of the course, we limit enrollment to 24 students who will work on the laboratories in pairs. We are requiring that the students purchase a book on MATLAB [6]. We distribute laboratory materials on a weekly basis in printed and World Wide Web formats. Next, we give the

* The authors would like to thank Texas Instruments for supplying five C50 DSP Kits and five 486 PCs; Intel for donating five Pentium PCs and ProShare packages; and AT&T for awarding a grant to develop the course. The authors would also like to thank the faculty of the EECS department for allocating resources to develop the course.

Week	Lecture
1	embedded systems & applications intro
2	sine waves: frequency, mag., phase, perception
3	sampling & aliasing, speech/music examples
4	linear and non-linear systems, LTI systems
5	filtering concepts: lowpass and highpass
6	filtering implementation: difference equations
7	digital representation: sampling/quantization
8	embedded digital system architecture
9	embedded digital system architecture
10	speech processing: pitch shifting
11	speech recognition
12	modulation: AM/FM
13	BFSK and matched filtering
14	image representation and processing
15	image decompositions, SVD-as-plaids demo
16	image enhancement

Table 1. Semester Lecture Schedule By Week.

Week	Tool	Discussion/Laboratory
1	none	none
2	MATLAB	matrices, MATLAB, tones
3	MATLAB	sampling tones, playback, aliasing
4	MATLAB	multiple tones, musical notes
5	MATLAB	vibrato, echo, tremelo effects
6	C50	C50 board/architecture intro
7	MATLAB	speech quantization
8	C50	tones: table lookup & diff. equ.
9	C50	multiple notes/tones in real-time
10	MATLAB	LPC coding
11	MATLAB	recognition of spoken digits
12	C50	DTMF codec
13	C50	BFSK modem
14	MATLAB	image filtering: median, texture
15	C50	processing of subblocks
16	none	none

Table 2. Semester Laboratory Schedule By Week.

course pre-requisites, equipment, personnel, and grading.

2.1. Pre-requisites

We require that the students take an introductory class on programming paradigms [11] (which uses Lisp) as well as a C or C++ programming class. This programming background makes it easier for the students to pick up the MATLAB language syntax and programming with arrays. The background in C helps them in learning the DSP assembly language. Optional but useful courses for the students to have taken include a class on machine structures and a class on differential equations. The machine structures background helps them understand DSP architectures, and the differential equations exposure helps them grasp difference equations. Students that have taken any of the junior signals and systems classes are excluded from taking this course.

2.2. Equipment

We provide the students with the equipment to sample and process speech, music, and image signals for the laboratory experiments. The students have access to four Intel Pentium PCs with built-in sound cards. The PCs are equipped with Intel's ProShare teleconferencing package which includes a video camera, video capture, and a microphone/earphone set. Through ProShare, the PCs are networked locally and to the Internet. Each PC also has a MIDI velocity keyboard. Full versions of MATLAB and SIMULINK are installed. For the embedded signal processing laboratories, each PC has a TMS320C50 DSP Kit and Evaluation Module from Texas Instruments Inc. Other available software includes Web browsers and a C++ compiler. Each station has a copy of several books and reference manuals [1, 2, 4].

2.3. Personnel

For the development and initial offering of the course, there are three professors, two teaching assistants, and three staff running the course. The professors are Edward Lee (signal processing, embedded systems, design methodology, and communications), David Messerschmitt (communications, signal processing, and VLSI architectures), and Shankar Sastry (controls and robotics). The teaching assistants are Ken Chiang and William Huang, who developed the laboratory exercises with general direction from the professors and staff. The staff are Brian Evans (MATLAB and Web course materials), Ferenc Kovac (equipment and scheduling), and John Reekie (TI DSP programming and board expertise).

2.4. Grading

We chose a grading system to reflect the laboratory emphasis of the class. Grades are based on an equal weighting of

- short quizzes,
- written laboratory reports, and
- oral laboratory reports.

The quizzes are given during the discussion section and last 15 minutes each. In both styles of laboratory reports, the students give short, qualitative explanations of their observations. The written reports are less than a page long, and the oral reports are 10 minutes or less in length.

3. COMPUTER MUSIC AND DIGITAL AUDIO

From this course, we hope that students gain an intuitive feel for basic discrete-time signal processing concepts, as well as an appreciation of the applications in which those concepts have been used. To this end, we are not placing the emphasis on the mathematical foundations of the course material, but instead on the reinforcement of qualitative concepts by hands-on laboratory work. When we introduce mathematical concepts, we appeal to the student's observation and intuition of physical phenomena.

After giving examples of sampled data in their daily lives, we introduce sampled signals by way of computer music [7]. We begin by discussing sinusoidal models for pure tones. By playing tones, the students hear the effect of changing magnitude, frequency, and phase on individual tones and on a sum of tones. In the next lecture, we present sampling and aliasing. We play properly sampled and aliased versions of the same speech and music signals to demonstrate how the harmful effects of aliasing are heard. In the corresponding laboratory, the students play a variety of tones to determine the frequency range of their own hearing, and undersample tones to hear aliasing.

For the next lecture, we introduce the concept of linear and non-linear systems, and the special case of linear time-invariant systems. We then play tones that have been processed by linear time-invariant, linear time-varying, and non-linear systems. The students can hear that the linear time-invariant system alters the amplitude but not the frequency of the tone. Since the ear is relatively insensitive to phase, the students are not able to distinguish the phase change in the single tone induced by the linear time-invariant system. For the linear time-varying system, we amplitude modulate the tone so the students hear the two resulting frequencies. We frequency modulate the tone to produce a rich set of tones for the example of a non-linear system. In the laboratory, the students experiment with representations of computer music. They play sequential tones to synthesize a bar of their favorite song, play multiple tones simultaneously, and modulate one tone with another. Time permitting, they can experiment with FM synthesis of musical tones.

Next, we introduce filtering from a qualitative point-of-view. We characterize filtering by passing certain qualities and rejecting others. We demonstrate the concept of lowpass and highpass filters by using tones and sampled waveforms. In the laboratory, the students code simple filters in MATLAB to produce a variety of simple digital audio effects, including vibrato, echo, reverberation, tremelo, and chorusing.

4. SPEECH

We began the course by focusing the application on computer music. In the context of computer music, the students experimented with sampling, aliasing, and filtering. We switch the application to speech processing to explore more about filtering, as well as signal quantization, coding, and interpretation.

In week #6, we introduce difference equations as a general framework to implement filters. We demonstrate that the complex exponential is a solution to difference equations. We cover the damped, oscillating, and underdamped cases by showing how the complex exponential behaves. At this point and throughout the course, we avoid using the z-transform. In the laboratory, the students get an introduction to the C50 boards and run several canned filtering demonstrations on them.

Next, we detail digital representation of signals on a computer by means of sampling and quantization. In the lecture, we play a speech signal quantized at various levels. To the surprise of the students, they find that speech quantized at one bit is actually intelligible. The students through hearing perceive the tradeoff between more bits and improved perceptual quality. In the laboratory, they discover the limit of perceptual improvement as they increase the number of bits. They also perform simple filtering operations on speech.

Now that we have introduced quantization, we spend the next two weeks talking about embedded digital system architectures, focusing on the C50 fixed-point DSP. In the first laboratory, the students generate tones by using table lookup and by using a difference equation on the C50 boards. We provide the routines to handle the input and output for them so they can concentrate on the algorithm. In the second laboratory, the students generate sequential tones and multiple tones in real-time on the C50. These two laboratories are in preparation for a dual-tone multiple frequency generator they build two labs from now.

The next two topics concern advanced speech processing topics of pitch shifting and speech recognition. In the first lecture, we discuss simple models of how speech is produced, and relate the models to difference equations. We discuss pitch and various ways to measure it. Then, we give an example of pitch-shifting by playing a Laurie Anderson CD. In the laboratory, the students run speech through a linear predictive coder (LPC). They are given the infrastructure to compute LPC coefficients. The students figure out how to window the speech and synthesize the same speech from the LPC model. Creative students use a variety of excitation models. In the second lecture, we introduce speech recognition, and in the laboratory, students implement pieces of a simple speech recognition system.

5. DIGITAL COMMUNICATIONS

Now that the students have seen representations of music and speech on the computer, we move into the issue of how to communicate this information. We present two lectures. The first is on AM/FM modulation, and borrows on the students experience tuning a radio station and selecting a television channel. We do not perform any noise analysis, but simply demonstrate how to communicate information carried by a modulating waveform. In the laboratory, the students leverage their previous laboratories on real-time tone generation to build a system that generates touch tones, i.e., a dual-tone modulated-frequency (DTMF) system [3, 9]. We also have them recognize the presence of a '1' digit in real time.

The second lecture is on binary frequency shift keying (BFSK) and its use in digital modems. We introduce the issue of matched filtering for this binary case. In the laboratory, the students reuse the DTMF codec from the previous laboratory to build a simple BFSK modem.

6. IMAGE PROCESSING

The students have now seen a variety of theory and applications of one-dimensional signal processing. We conclude the course by having the students learn how to extend their knowledge into two dimensions by way of image processing. We begin with representations of images on the computer (as rasters and matrices) and how to process them. We show that the filtering concepts generalize to images. In the laboratory, they use MATLAB to extract the edges and texture in the image and to remove salt-and-pepper noise. They have to figure out which of a lowpass, highpass, and median filter to use to accomplish these tasks.

Next, we discuss image decompositions. We demonstrate predictive coding and singular-value decomposition. The visualization of singular-value decomposition in two dimensions is a combination of plaids (i.e., weighted sums of products of column and row vectors). As the number of plaids (terms) increases, the decomposed image approaches the original, as shown in Figure 1. In the laboratory, the students explore predictive coding to implement one of the

JPEG schemes in real-time by processing one *8 x 8* subblock at a time. We conclude the course with a lecture on image enhancement.

7. CONCLUSION

Until a decade ago, many if not most of the entering electrical engineering students had background with some form of analog circuitry, e.g., building radios or working on cars. Today, however, the student entering electrical engineering and computer science is far more likely to be from a computer background and more accustomed to a digital world. Based on this observation, the Georgia Institute of Technology [12] introduces electrical engineering to their computer engineering students by means of a sophomore discrete-time systems class.

We are implementing our own sophomore discrete-time systems course that mixes signal processing and communications with computer science and engineering. Our goal is to give the students an intuitive and practical understanding of crucial concepts in discrete-time systems such as the frequency domain, sampling, aliasing, and quantization. The students test concepts in the familiar world of digital computers by programming desktop processors for simulation and embedded processors for real-time implementations.

REFERENCES

[1] The MathWorks Inc., *The Student Edition of MATLAB Version 4 User's Guide*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[2] *TMS320C5x User's Guide*. Texas Instruments, Inc., 1993.

[3] M. A. Chishtie, ed., *Telecommunications Applications with the TMS320C5x DSPs*. Texas Instruments, Inc., 1995.

[4] The MathWorks Inc., *The Student Edition of SIMULINK User's Guide*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[5] E. A. Lee, "Signal processing experiments using Ptolemy — instructor's manual." (contact the author at `eal@eecs.berkeley.edu`), May 1994.

[6] D. Hanselman and B. Littlefield, *Mastering MATLAB*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1996.

[7] V. Stonick and K. Bradley, *Labs for Signals and Systems Using MATLAB*. Boston, MA: PWS Publishing Inc., 1995.

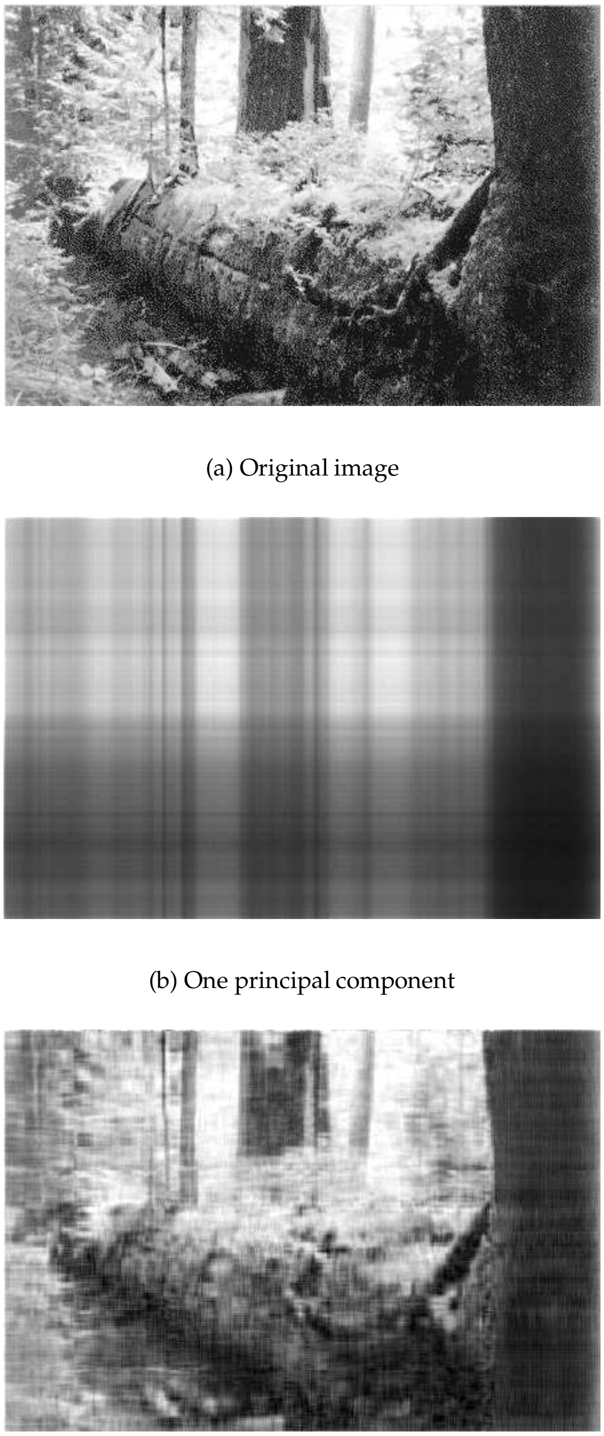
[8] C. S. Burrus, J. H. McClellan, A. V. Oppenheim, T. W. Parks, R. W. Schafer, and H. Schüssler, *Computer-Aided Exercises for Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1994.

[9] K.-S. Lin, ed., *Digital Signal Processing Applications with the TMS320 Family*, vol. 1. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[10] D. L. Jones and T. W. Parks, *A Digital Signal Processing Laboratory Using the TMS32010*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1988.

[11] H. Abelson and G. Sussman, *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press, 1985.

[12] V. K. Madisetti, J. H. McClellan, and T. P. Barnwell, "DSP design education at Georgia Tech," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, vol. 5, (Detroit, MI), pp. 2869–2872, May 1995.



(a) Original image

(b) One principal component

(c) Twenty principal components

Figure 1. Illustrating the tradeoff of compression rate vs. quality in a data-dependent lossy compression algorithm that sums up a finite number of plaid patterns generated by the principal singular-value components of the image treated as a matrix.

REAL-TIME DSP FOR SOPHOMORES

Kenneth H. Chiang
Edward A. Lee , *Brian L. Evans*
David G. Messerschmitt , *William T. Huang*
H. John Reekie and *Ferenc Kovac*
Shankar S. Sastry [1]
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720-1770
E-mail: *eecs20@hera.eecs.berkeley.edu*
WWW: http://www-inst.eecs.berkeley.edu/~ee20

We are developing a sophomore course to serve as a first course in electrical engineering. The course focuses on discrete-time systems. Its goal is to give students an intuitive understanding of concepts such as sinusoids, frequency domain, sampling, aliasing, and quantization. In the laboratory, students build simulations and real-time systems to test these ideas. By using a combination of high-level and DSP assembly languages, the students experiment with a variety of views into the representation, design, and implementation of systems. The students are exposed to a digital style of implementation based on programming both desktop and embedded processors.