A Vertically-Integrated Application-Driven Signal Processing Laboratory

Lisa G. Huettel and Leslie M. Collins

**Department of Electrical and Computer Engineering,
Duke University, Durham, NC**

Abstract

Hardware-based laboratories have been successfully integrated into individual Digital Signal Processing (DSP) courses at many universities. Typically, most hardware-based DSP laboratory experiences are offered to upper-level students and focus on programming the signal processor. Although fundamental concepts are explored in laboratory exercises, the emphasis often remains on the mechanics of hardware implementation. Thus, topics are not presented in the context of realistic applications. While such an approach may be ideal for preparing motivated upper-level students for future careers in signal processing, it is not suitable for students with no prior experience in the field. The signal processing laboratory being developed at Duke University is modeled, in part, after existing successful signal processing laboratories, but introduces two innovative features. First, the new laboratory will be integrated into multiple courses from the sophomore to senior level, rather than a single course. Second, the laboratory exercises will be application-driven and will emphasize the development of signal processing algorithms to be implemented on the hardware. As the students advance through the signal processing curriculum, they will transition from high-level algorithm generation to hardware-level design and implementation. This hierarchical training will provide a thorough, extended, and increasingly focused exposure to signal processing.

1. Introduction

Digital signal processing (DSP) is central to modern Electrical and Computer Engineering (ECE) undergraduate curricula. The discipline of signal processing combines an extensive mathematical background with practical design skills. To prepare for a successful career in signal processing, whether in industry or academia, students should develop expertise in two domains: the theoretical understanding of signal processing problems and the design of devices or algorithms to solve those problems. As part of its ongoing curriculum reform, the ECE department at Duke University has implemented a new DSP laboratory that impacts student instruction in multiple courses. In this paper, we describe the motivation for creation of our DSP laboratory, its pedagogical principles, its implementation in our undergraduate major, and the key challenges that remain.

Prior to the implementation of our new laboratory in Fall 2004, Duke ECE undergraduates received instruction in DSP principles within two courses. In the required sophomore-level *Signals and Systems* course (ECE 64), students learn about introductory signal processing concepts like frequency domain representation, sampling, and modulation. Complementing the primary lecture instruction are several student projects, including a music synthesis exercise that requires students to use MATLAB (Mathworks, Inc.) to generate a musical selection of their choice. When creating their musical excerpts, all students learn the basic concepts of signal generation and manipulation, sampling, and frequency harmonics, while the more advanced students incorporate signal processing techniques like modulation to enhance their composition. Other projects include a very basic MATLAB simulation of source detection in which the students study correlation, a simple modulation simulation done in MATLAB, and the construction and testing of a working AM radio. In the junior/senior level elective *Fundamentals of Digital Signal Processing* (ECE 180), students who are particularly interested in signal processing learn more advanced DSP concepts, including transform analysis and digital filter design. The students are again given MATLAB programming assignments, so that they become proficient in the use of MATLAB commands for signal processing.

While these courses provided students with a strong grounding in signal processing theory, they had only limited practical components. All student projects used MATLAB simulations on stand-alone UNIX systems, without measurement or digital signal processing (DSP) components. There was no design course for senior students specifically interested in DSP, nor was there a physical laboratory that had the equipment necessary to support a broad range of independent student projects. These limited resources do not motivate our undergraduate students to pursue careers in signal processing, nor do they prepare students for the development environments they are most likely to encounter after they leave Duke. In order to retain current students, to produce well-prepared students that are competitive in the marketplace, and to continue to attract the highest quality students, our faculty have developed new laboratory resources for DSP instruction.

Our goal is to develop a signal processing laboratory that will serve sophomore- through senior-level students in core, topic, and design courses. Many institutions have created DSP laboratories for their ECE students, and we have adapted material that has been successfully used elsewhere. In typical DSP laboratories, upper-level students gain direct experience with programming DSP chips in order to solve engineering problems. However, our planned laboratory has two innovative aspects. First, we will integrate the laboratory vertically throughout the curriculum, to enhance or replace existing simulation exercises in both introductory and advanced signal processing courses. This approach provides students with early, hands-on experience with industry-standard hardware and software technology, as students will use MATLAB, Simulink, and LabView in conjunction with DSP hardware and test and measurement equipment. Second, we have adopted an application-based approach. As student feedback has indicated, application-driven exercises enhance the appeal of signal processing without compromising learning of theoretical concepts or precluding more specialized instruction in DSP design and programming.

In the following sections, we detail our initial implementation of the DSP laboratory in Fall 2004, including equipping a laboratory room and using application-driven exercises in our upper-level *Fundamentals of DSP* course.

2. Pedagogical Principles

Hardware-based laboratories have been integrated into *individual* Digital Signal Processing courses at many universities[1-6]. Such laboratories typically include exercises that familiarize students with the hardware and programming requirements of the DSP chip, covering topics such as hardware interrupts and Assembly Language programming. Although concepts such as FIR/IIR filtering and sampling are also covered in laboratory exercises, the emphasis often remains on the mechanics of hardware implementation so that these topics are not presented in the context of realistic applications. While such an approach may be ideal for senior-level students, it is not suitable for less-experienced students, as the perceived technical difficulty of such a course may deter students who do not immediately perceive the relevance of the material[7]. In a pedagogical model developed by Wright et al.[7], interactive demonstrations, MATLAB simulations, and real-time DSP programming are used to supplement the theory presented to the students. Their experience indicates that this approach results in a more thorough understanding of DSP topics, making DSP more accessible to all students. The laboratory at Duke is based on the typical hardware-based signal processing laboratories and incorporates the pedagogical model of Wright et al. [7], but has been modified by introducing two innovative features.

2.1 Application-driven exercises

In many existing courses (including prior courses in our program), topics are explored in a relatively abstract manner. For example, students might explore filtering by creating a FIR filter to meet certain specifications (e.g., "Design a length-20 low-pass filter that has a cut-off frequency of 2kHz using the windowing method.") and then testing its output spectrum for a variety of inputs. In an alternative approach, as will be implemented in the proposed laboratory, students could be given a design problem (e.g., design a touch-tone simulator or filter out a constant or changing noise source from a musical recording) for which they would need to implement an FIR filter. Both approaches would allow students to explore the consequences of changing the filter length, design method, or other parameters. But by linking the theory to a real application and by allowing students to determine and test their own filter specifications, the second approach is more likely to stimulate students' interest in the material.

The laboratory exercises will illustrate fundamental signal processing concepts using real-world examples and applications, thus complementing theoretical material presented in the classroom and MATLAB simulations assigned for homework. There are a number of sources for applied laboratory experiments which can be modified to meet the goals of our laboratory[2,4,9]. The following table lists possible applications for laboratory exercises and the key concepts illustrated or explored in each exercise.

Table 1. Examples of DSP applications for the newly developed laboratory.

Application	Summary of Exercise	Key Concepts
Music Synthesis	Students will investigate the difference between sounds created by various musical instruments and synthesize the notes produced by real instruments.	Mathematical description of signals, periodicity, harmonics, Fourier series
Cochlear Implant Speech Processor	Students will learn the basic operation of the speech processor used with a cochlear implant and will design and test their own processor.	FIR and IIR filtering, frequency analysis, spectrograms, FFT, windowing
Landmine Detection	Students will develop an algorithm that can distinguish the response of a metal detector to a landmine from its response to clutter objects.	Correlation, matched filtering
Voice Scrambler/ De-scrambler	Students will design and implement a system that scrambles an audio signal (generated by the student), transmits it to another location, and de-scrambles it in real-time.	Modulation and demodulation, filtering, sampling
Noise Cancellation	Students will design and implement an algorithm capable of filtering out a steady or variable source of noise effectively enough that the student's voice can be clearly understood.	Bandpass filtering, adaptive filtering, LMS algorithm, spectrum analysis
Musical Special Effects	Students will design and implement systems to simulate audio effects such as echo, chorus, and vibrato and apply these to their own voices.	FIR and IIR filtering
Digital Audio Recording	Students will analyze the quality of digital recording of speech and music using various digitizing parameters.	Sampling, aliasing, anti-aliasing filters
Analysis of Biosignals	Students will acquire and analyze biosignals (e.g., their pulse rate before, during, and after strenuous activity) and design an alarm system that indicates when the signal exceeds a preset threshold.	Sampling, aliasing, frequency analysis, FFT, frequency resolution, anti-aliasing filters

Many of the applications will be drawn from the ongoing research of Duke faculty, including the detection of buried landmines, voice recognition, medical image processing, and bio-signal monitoring. For example, in one exercise drawn from the research of one of the authors, students will listen to simulated speech signals generated by the speech processor of a cochlear implant. Students will learn about the functioning of a healthy cochlea, which behaves like a bank of

overlapping band-pass filters, as well as about how the implant technology is limited in the number of electrodes that can be implanted (i.e., reducing the effective number of filters). Students will then design their own speech processor according to these physiological constraints. Students will visually compare speech spectrograms to electrodograms generated by their processor and will aurally compare the original speech to the simulated speech sounds. This hands-on application provides a natural transition to independent study or team design projects in the research laboratory of the relevant faculty member.

Laboratory exercises will be designed to ensure synergy between the hardware- and software-based components of an experiment and to incorporate concepts learned in previous courses. For example, in the "Noise Cancellation" laboratory, students will explore filter design (fixed filters in the introductory course, adaptive filters in the advanced course) by comparing how theoretical, analog, and digital filters process the same audio signal. The students will be given a music file corrupted by noise. Drawing on their previous circuit design experience, they will then design and construct an analog filter to remove the noise. Next, the students will design a comparable digital filter using MATLAB, plot the theoretical frequency response, implement the filter on the DSP board, and graph the actual frequency response. Students will compare features such as the passband ripple, stopband attenuation, and rolloff for each filter. Students can also pursue some perceptual experiments by listening to the outputs of the various filters, and comparing their perceived quality to a variety of error metrics for predicting sound quality.

2.2. Vertical curriculum integration

The second innovative feature of our approach will be the integration of the laboratory into multiple courses from the sophomore to senior level, rather than a single course. Students will progress from the development of signal processing algorithms (to be implemented on the chip) to programming the chip directly. This approach will particularly benefit students in introductory courses who do not have the theoretical or technical background necessary for direct programming of DSP chips. Recently developed software programs (such as the *MATLAB Link for Code Composer Studio* and *Embedded Target for TI C6000 DSP*; Mathworks, Inc.) minimize this obstacle by allowing students to implement designs on DSP chips using more familiar and/or intuitive software tools such as MATLAB and Simulink. This approach allows even beginning students to gain hands-on experience with DSP hardware and equipment at the same time they learn the theoretical fundamentals of signal processing[7,8]. Additionally, by integrating the laboratory into both lower- and upper-level classes, we will be able to introduce design principles into the curriculum at an early stage, improving the intellectual development of a large number of students.

Our two introductory signal processing courses (the required *Signals and Systems* and the elective *Fundamentals of DSP*) will focus on the signal process*ing*, rather than the signal process*or* and will not require an in-depth understanding of the DSP hardware or programming itself. Rather, hardware implementation and direct programming of the DSP will be the focus of a new senior-level design course. This vertically integrated approach has the advantage that by the time a student reaches the design course, he or she will already be familiar with the hardware and software and will be able to pursue a more advanced, semester-long design project. As students transition from software-based design to direct programming, they will be exposed to

the realistic development environment they will encounter if they pursue a career in signal processing.

3. Creating a DSP Laboratory

The authors were awarded a Course, Curriculum, and Laboratory Improvement (Adaptation and Implementation) grant from the National Science Foundation in 2004. As existing signal processing courses at Duke University utilized MATLAB simulations and had no physical laboratory, the first step was to procure the necessary hardware and software for the planned laboratory. This was done, as described below, prior to the Fall semester of 2004. In that semester, the first realization of the new DSP laboratory was incorporated into the *Fundamentals of DSP* course. Examples of laboratory exercises are described below.

3.1 Physical Laboratory Environs

The laboratory has 12 stations each composed of 1) a PC with a microphone, speakers, and headphones and 2) test and measurement equipment, including a function generator, a digital oscilloscope, a multimeter, and a power supply. For our purposes, it was important to select a DSP board that could easily interface with MATLAB and SIMULINK to minimize the perceived or real technical difficulties a beginning student might encounter if he or she had to program the chip in Assembly or C. Not only did the Texas Instruments' TMS320C6713 DSP Starter Kit (DSK) satisfy this criterion, but TI also expressed an interest in the development of our laboratory and provided hardware through their University Program. Software available on each PC includes a C Compiler/Assembler/Linker, debugger, and simulator; MATLAB 7 with Signal Processing, Image Processing, Data Acquisition, and Filter Design Toolboxes (The Mathworks, Inc.); SIMULINK with DSP and Communications Blocksets (The Mathworks, Inc.); MATLAB Link for Code Composer Studio and Embedded Target for TI C6000 DSP (The Mathworks, Inc.); Real-Time Workshop (The Mathworks, Inc.); and Goldwave Digital Audio Editor.

3.2 Example Laboratory Exercises

The following subsections describe three of the exercises completed by students in the Fall 2004 semester of *Fundamentals of Digital Signal Processing*. All of these students had taken the introductory signal processing course, *Signals and Systems*, and were familiar with MATLAB programming. The students had no prior experience with DSP hardware, although in future semesters the students will have already used the DSP hardware and software in the previous *Signals and Systems* course. Thus, because this was the first offering of a signal processing course with a hardware-based laboratory, some time had to be devoted to familiarizing the students with the available tools. Having done this, the students were able to complete several challenging design problems, as described below.

3.2.1 Introducing Students to DSP Hardware and Software

As none of the students in the course had experience with DSP hardware and few were comfortable with C or Assembly programming, an introductory laboratory was needed to

familiarize the students with the TI 6713 DSK and the way in which SIMULINK and MATLAB could be used to design and download algorithms onto the DSP chip.

The objectives of this introductory project were for students to:

1. Become familiar with the hardware and software that will be used in the laboratory
2. Build a template for communication between SIMULINK and the DSK to be used in all future projects
3. Build a basic system in SIMULINK, then download and run this system on the board
4. Experiment with audio processing using SIMULINK and the DSK
5. Learn how to use multiple signal sources (function generator, saved files, microphone) as input to the DSK
6. Learn how to display and analyze system output (headphones/speakers, oscilloscope, saved files, MATLAB)
7. Build a more complex system in SIMULINK using pre-defined filters

These objectives were achieved by progressing through a series of four mini-projects, each building upon the previous one. For the first mini-project, a template for communicating between SIMULINK (the primary system design environment) and the DSK was designed. Although many of the settings were not intuitive, step-by-step instructions for setting the parameters were provided to the students. This template allowed students to design and implement their algorithms using MATLAB and SIMULINK, rather than writing their code in C (which is the language used with Code Composer Studio, the interface included with the DSK). The second mini-project required the students to design and implement a system to control the on-board LEDs using the on-board DIP switches. By doing this, the students began to explore the built-in commands, or 'blocks', of SIMULINK. Next, the students built a simple audio processor and explored the variety of input and output options (e.g., microphone or function generator inputs; speaker, oscilloscope, or recorded file outputs). Finally, the students were given the block diagram of a more complex system which produced various audio effects and implemented it in SIMULINK. By the end of this exercise, students were comfortable with SIMULINK's block diagram approach to implementing a system and had begun to explore the variety of block functions available in SIMULINK. In addition, they had explored many of the input and output options available, which would be important for future projects for which the students had to devise ways of testing and debugging their systems.

3.2.2 Application Example I: DTMF system

In a second laboratory project, students designed and implemented a dual-tone multi-frequency (DTMF) encoder/decoder. An example of a DTMF system is a touch-tone phone: when any key on the telephone key pad is pressed, a pair of sinusoids are generated and added together, producing a "dual tone." Each key is associated with a unique pair of tones and only seven, harmonically unrelated tones are needed to encode the digits 0 – 9. The decoder identifies which key has been pressed by analyzing the dual tones, extracting the two frequencies, and looking up the corresponding digit. Background information that described the principles behind DTMF and the touch-tone phone system application was provided, both to motivate the students and to provide the parameters and constraints with which they had to work.

The objectives of this project were for students to:

1. Gain an understanding of the standard DTMF phone dialing system
2. Design and implement a DTMF tone generator (encoder)
3. Learn how to use the TI DSK to generate sinusoidal signals
4. Design and implement a DTMF receiver (decoder)
5. Design simple FIR bandpass filters to meet desired specifications
6. Study the tradeoffs between filter order and frequency response

Students achieved these objectives by designing and implementing the two major subsystems: a DTMF encoder and a DTMF decoder. The students were instructed to design the encoder in such a way that the tone generated was determined by the position of the on-board DIP switches, thereby enabling real-time operation of the system. To achieve the proper system behavior, the students had to design and encode an algorithm using SIMULINK which would instruct the DSK to (1) read the input from the DIP switches, (2) translate the DIP switch input into the proper pair of tones, and (3) generate a DTMF tone output.

The students were guided to approach this design problem by breaking it down into small steps that could be easily tested for proper behavior. Thus, students began by designing a model to generate two tones at desired frequencies and sum them. In generating these tones, students considered parameters such as the sampling rate and the buffer size. Proper operation was verified aurally and visually, using an oscilloscope.

The next step was to extend the model such that all digits 0 – 9 could be represented with pairs of tones and to design a system to translate DIP switch positions into the corresponding DTMF tone pair. The underlying subsystems in this model (tone generation and DIP switch input) were simple and had been explored in the introductory laboratory. However, combining them into a functional system proved challenging for the students.

For the decoder, the students designed a system that received the tones generated by the encoder (and transmitted from one DSK to another via audio cables) and translated them back into binary numbers which were displayed using the on-board LEDs. This could be implemented using two subsystems: a band-pass filterbank and a detector. To construct the filterbank, the students first used MATLAB to design a set of length-40 FIR filters that isolated the individual frequency components comprising the transmitted signal. They then plotted all of the filters on a single graph and speculated whether this filterbank would be sufficient to decode the input signal. They observed that some tones fell within the passband of multiple filters. Thus, some ambiguity was possible, depending on the design of the detector. Thus, the students determined the desired specifications of the filter and then used that information to design and analyze their detector. This exercise required them to explore tradeoffs between filter length and bandwidth for FIR filters. Once an acceptable set of filters had been designed, the filter coefficients were imported into the SIMULINK model of the DTMF decoder.

The second subsystem in the decoder is a detector that determines which frequencies are present in the received signal, based on the strength of the signal at the output of each band-pass filter. In a real system, this scoring process is critical because noise and interference can distort the

transmitted signal. However, the system designed by the students was essentially noise-free. Thus, assuming the filterbank subsystem had been correctly designed such that each tone was passed by only one filter, the detector simply had to determine which two filters had the largest outputs and map the pair of tones to a digit.

This project presented several challenges to the students. To begin with, the students had to think the problem through at a relatively high level, focusing on the algorithm itself. At this stage, they realized they needed to use the DIP switches to control the generation of tones which, after being transmitted, had to be passed through a filterbank in order to decode the signal. The next challenge was to use their theoretical knowledge to design the actual components of the system. As this was a real-world application, the students were able to extract the parameters of their system (e.g., center frequency and bandwidth of the filters) directly from the problem statement. This approach is clearly preferable to a situation where the students are simply given the parameters as it requires them to approach the problem in a more realistic way and to develop a fuller range of design skills. The SIMULINK approach successfully shifted the focus of the exercise from debugging C or Assembly code and enabled the students to focus on their algorithm development.

3.2.3 Application Example II: Pulse Rate Monitors

In the final project of the semester, students designed and implemented a system that could monitor their pulse rate and sound an alarm if this rate exceeded a pre-set threshold. In addition to their work in the laboratory, the students were asked to collect data in the field and bring it back to the laboratory. For this task, students were given 1) a piezo-electric transducer designed to measure pressure pulses, such as those generated in a finger by a beating heart, and 2) an Apple iPod to which the transducer could be connected. This project not only motivated students through a real-world application of many of the concepts covered in class, but it also gave meaning to many of the parameters used in their system design (e.g., filter specifications) since the application was one to which they could personally relate.

The objectives of this project were for students to:

1. Learn how to use external sensors (e.g., pressure sensor) as input devices to the board
2. Capture and store data collected using external sensors
3. Record and download non-audio data using the iPod
4. Design and test a system that meets given specifications
5. Estimate the frequency of a signal as it changes over time
6. Design and implement a system that performs basic signal conditioning
7. Collect non-audio field data using the iPod
8. Analyze real data and to discover issues that arise when processing real data
9. Design and test a pulse rate monitoring system

This project once again built on earlier laboratory projects, with many of the subsystems having been implemented previously. At this point in the semester, the students were quite comfortable using SIMULINK and MATLAB to design models which could be compiled, downloaded, and run on the DSK. This allowed them to focus on the development of their algorithm. The use of a

real-world application had two distinct advantages. First, the students were able to collect and analyze their own data. From a motivational standpoint, this was very important and substantially increased the students' interest in the project. Second, as the students were directly involved in every aspect of the data collection and system design and testing, they were better able to relate the time- and frequency-domain waveforms they observed to the physical phenomena that cause them. Their improved understanding of how the data was generated guided their approach and enhanced their understanding of how the data should be processed.

The students began by experimenting with the transducer, observing how its placement affected the quality of the signal. This was their first significant introduction to signals with noise and the students quickly realized, after looking at the time-domain waveform, that they would have to design a system that could extract a pulse rate from a noisy signal. Realizing that the range of reasonable pulse rates was quite small and had both upper and lower bounds, students quickly came to the conclusion that much of the noise could be eliminated by using a band-pass filter. Through this project, students gained experience in filter design, transform analysis, and general system design. They were forced to consider real tradeoffs, such as filter order versus sharpness and memory requirements, and the effects of signal buffering on computation time and delay in the system.

One concept for which the project had a dramatic effect upon students' understanding was the relation between the length of the discrete Fourier transform (DFT) and the resulting frequency resolution. In their system, the students calculated the pulse rate by locating the sample number, N_{max}, at which the largest peak in the frequency spectrum occurred. This value could be directly related to the actual pulse rate if the length, L, of the DFT and the sampling frequency of the signal, f_s, were known (pulse frequency = $N_{max}*f_s/L$). In addition, students determined the DFT length that allowed them to measure reasonable changes in the pulse rate. These relatively simple calculations, which they had previously encountered in more abstract homework problems, became meaningful to the students when applied to a real problem.

4. Key Challenges Remaining

As our laboratory revisions continue, we will face new implementation challenges. The three projects described above for our upper-level *Fundamentals of Digital Signal Processing* course form the skeleton of a full DSP laboratory, but they must be complemented with additional projects that flesh out the topics covered in lecture. As these new projects are developed, we plan to transition most of them to the introductory *Signals and Systems* course, so that students complete relevant projects upon their entry to the signal processing track in the Duke ECE curriculum. We will replace those simpler projects in the upper-level course with more advanced projects, greatly improving the depth of our topical coverage – this will only be possible once a cohort of students has passed through the revised introductory course. We will next create novel design courses, through cooperation between signal processing and computer engineering faculty within our department, that will provide new opportunities for advanced/independent study. Finally, we need to ensure comprehensive and timely integration with our ongoing curricular revision that provides an overarching, application-based approach to ECE education.

Bibliography

1. R.G. Baraniuk, C.S. Burrus, B. Hendricks, G. Henry, A. Hero, D. Johnson, D.L. Jones, J. Kusuma, R. Nowak, J. Odegard, L.C. Potter, and K. Ramchandran (**2002**). "Connexions: DSP education for a networked world," *2002 Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 4144-4147.
2. The Connexions Project, cnx.rice.edu
3. D.L. Jones (**2001**). "Designing effective DSP laboratory courses," *2001 Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 2701-2704.
4. D.E. Melton, C.J. Finelli, L.M. Rust (**1999**). "A digital signal processing laboratory with style," *1999 Proc. ASEE/IEEE Frontiers in Education Conf*, Session 12b6, November, 1999.
5. M. Nahvi (**1999**). "Real-time digital signal processing design projects in an undergraduate DSP course and laboratory," Texas Instruments DSPS Fest 1999.
6. E.A. Lee (**2000**). "Designing a relevant lab for introductory signals and systems," IEEE Signal Proc. Education Workshop, October, 2000.
7. C.H.G. Wright, T.B. Welch, D.M. Etter, M.G. Morrow (**2002**). "Teaching DSP: Bridging the gap from theory to real-time hardware," *Proc. Am. Soc. for Eng. Education*, Session 3220, June, 2002.
8. M.G. Morrow, T.B. Welch, C.H.G. Wright (**2000**). "An inexpensive software tool for teaching real-time DSP," IEEE Signal Proc. Education Workshop, October, 2000.
9. R. Chassaing (**2002**). *DSP Applications Using C and the TMS320C6x DSK*, (John Wiley & Sons, New York).

Biographical Information

LISA G. HUETTEL, Ph.D.**,** is an Assistant Professor of the Practice and Director of Undergraduate Laboratories in the Department of Electrical and Computer Engineering at Duke University. She is interested in the application of statistical signal processing to remote sensing and engineering education. She received her M.S. and Ph.D. in Electrical Engineering from Duke University.

LESLIE M. COLLINS, Ph.D., is an Associate Professor in the Department of Electrical and Computer Engineering at Duke University. Her research interests lie in physics-based statistical signal processing with applications in remote sensing and auditory prostheses. She received her Ph.D. in Electrical Engineering: Systems from the University of Michigan.