Teaching DSP: Bridging the Gap from Theory to Real-Time Hardware

Cameron H. G. Wright
Department of Electrical Engineering
U.S. Air Force Academy, CO

Thad B. Welch, Delores M. Etter
Department of Electrical Engineering
U.S. Naval Academy, MD

Michael G. Morrow
Department of Electrical and Computer Engineering
University of Wisconsin-Madison, WI

Abstract

Many digital signal processing (DSP) topics are difficult for undergraduates to internalize, but studies have shown that demonstrations and laboratory experiences can facilitate the process. In the past, many barriers prevented including real-time DSP hardware in an undergraduate curriculum. This paper describes a pedagogical model the authors have developed which includes theory, demos, lab exercises, and real-time DSP experience using MATLAB, C, and real-time DSP hardware that overcomes the barriers. This model has been very successful.

1 Introduction

A common complaint heard from electrical engineering (EE) undergraduates is that many (if not most) of the EE topics are difficult to visualize. One of the fastest growing fields in EE, digital signal processing (DSP) certainly has more than its share of concepts that fit this description. In particular, making the leap from MATLAB DSP simulations to real-time DSP hardware has proven to be singularly challenging for faculty and students alike. It is well known that demonstrations and laboratory experiences help most students internalize both the theoretical underpinnings and the practical ramifications of various DSP topics,[1–3] but real-time DSP hardware and software have usually been considered too difficult for undergraduates. This high degree of difficulty is due to many factors, including the need to understand parallel processing, multiple memory busses, specialized instruction sets, and—most importantly—a lack of documentation that is "readable" by the non-expert.

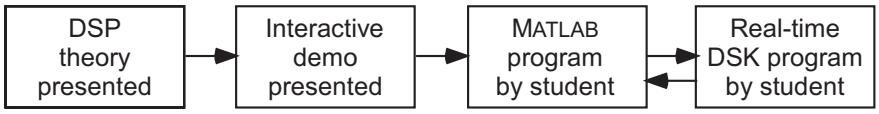

Figure 1: A systematic model for teaching DSP. The last two steps can be iterated as needed.

Over the last few years, we have developed a systematic method to teach DSP to undergraduates. It provides students with a firm bridge from their first exposure to theory all the way to practical implementation of real-time DSP code on industry-standard hardware, such as the Texas Instruments (TI) C6711 digital signal processing starter kit (DSK). Previous articles[4–11] have described the application of this method to specific DSP concepts; this article generalizes the lessons learned and outlines the overall method in such a way that it could be applied to any DSP topic. A world wide web URL is provided at the end of this article for downloading the software that is a key component of our method for "bridging the gap" from theory to real-time hardware.

2 A Graduated Approach to Teaching DSP

2.1 Using a "Bridge" to Real-Time

A systematic model for teaching DSP is shown graphically in Figure 1. It begins with the traditional presentation of the theory behind each new topic, followed by a specific progression of exercises.

To facilitate the learning process in a DSP class, demonstrations of fundamental topics are helpful supplements to the theory. In particular, computer-based demonstrations are highly effective for a student's initial grasp of a new DSP topic; this is reflected in the content of many newer DSP texts.[12–17] We take advantage of the fact that the software package MATLAB[18] and its related toolboxes have become a mainstay in most EE programs. Given our students' familiarity with MATLAB, computer exercises that implement DSP theory are a natural approach. However, today's students are quickly bored with a "canned demo," and the application of these demonstrations to real-time DSP is limited. In response we have created a series of interactive demos that allow the student to "play" with a concept and engage in "what if?" explorations, while laying the foundation for real-time applications.

Such interactive demonstrations typically result in much greater comprehension of the topic by the student, yet we have found that taking the next step of requiring the student to program an example of the concept in MATLAB is needed to solidify the student's understanding. DSP programming has long been used in graduate-level DSP classes, but has only recently been applied to undergraduates. When the student can comfortably create a basic MATLAB program that performs a particular DSP operation (such as FIR filtering) on stored

data, then we can start to ease them into real-time DSP.

Moving beyond a MATLAB-only program to a real-time hardware implementation is highly desirable from a pedagogical point of view. Many practical issues and learning opportunities occur only when the students try to adapt their newly acquired knowledge to the challenge of real-time DSP. For example, interrupt-driven processing, tradeoffs of on-chip versus off-chip memory access, and utilization of specific hardware capabilities are all issues that arise only when the DSP operation is implemented in real-time hardware. In the past, this next step in understanding has been impeded by a *very* abrupt transition, in terms of cost to equip a student laboratory and in terms of the steep learning curve (for both students *and* faculty) of unfamiliar systems and software. We therefore developed a software and hardware "bridge" between MATLAB and real-time DSP hardware that makes it possible to smoothly and incrementally transition from the MATLAB-only domain to a full hardware implementation operating in real-time, while retaining as needed the impressive capabilities of the MATLAB display engine. Using this approach, students are able to develop and enhance their own real-time DSP programs in an iterative way, "moving" more and more of their code from the realm of MATLAB over to C or assembly language for the DSK. The last two steps of Figure 1 illustrate this iterative nature of the model.

2.2 Choosing the DSP Hardware

As detailed in previous articles, we chose to construct our DSP educational platform around MATLAB and the TI C6x DSK. The current version of the TI C6x DSK makes use of the VLIW architecture TMS320C6711 microprocessor and includes 16 MB of memory, basic support circuitry, and excellent software development tools (Code Composer Studio) that include an optimizing C compiler, debugger, assembler, and linker. This meets our criteria of low cost, sufficient processing power, ample memory, and a versatile software development environment. Furthermore, while other companies such as Analog Devices and Motorola also manufacture DSP microprocessors, we have been unable over the years to elicit interest from any other company in the educational segment; only TI has consistently demonstrated such interest. See reference [19] for details on this DSK, and see reference [20] for more information on TI DSP products and support in general.

Unfortunately, to keep costs down, the native codec on the C6711 DSK board is the TI TLC320AD535 chip, a single-channel telephone-quality device with a maximum sampling frequency of only $f_s \approx 8$ kHz. This severely limits the utility of the DSK for a variety of DSP applications we desire for our students. However, we feel the DSK's advantages outweigh this disadvantage, and the inclusion of a flexible Expansion Daughter Card Interface on the DSK allows us to circumvent this limitation quite easily. We supplement the basic DSK as needed for a particular application with more capable I/O such as a CD-quality stereo codec board[21] or a high speed multichannel ADC.[9,11] These small daughter cards are available to the public at very low prices; see reference [22] for more information. Using the C6711 DSK and the appropriate daughter card as the core, a professor can populate a highly flexible real-time DSP student laboratory at low cost.

2.3 Floating-Point or Fixed-Point?

One of the primary choices in practical DSP hardware today is the question of floating-point versus fixed-point implementations.[23] While the floating-point ability of the TI C6711 DSK offers a pedagogical advantage (topics such as scaling and overflow may be postponed until later), the C6711 processor can also run fixed-point code if the professor desires. This "two for the price of one" ability represents another strong advantage of the C6711 DSK, in our opinion.

3 Examples Of Using The Model

To clarify the use of this teaching model, we describe three examples of DSP topics that have benefitted from this approach: basic FIR filter design, an application of a particular type of FIR filter (the Hilbert transformer) for communications systems, and audio special effects (flanging and chorus).

3.1 FIR Filter Design

One of the first "theory to real-time implementation" labs that our students experience is the design of an FIR filter. Our students learn the basic theory first, such as the generalized transfer function given by

$$H(z) = \sum_{i=0}^{M} h[i] z^{-i} \tag{1}$$

where M is the filter order and $h[i]$ is the i^{th} coefficient of the filter's impulse response.[24]

We use interactive demos, based first upon the `sptool` GUI provided with MATLAB's Signal Processing Toolbox shown in Figure 2. Students are encouraged to use `sptool` to create FIR filter designs, but also introduced to the underlying programs such as `remez` which can produce filters `sptool` cannot. Next, we move them to a program we designed called `qfilt` (see Figure 3) which serves several purposes.[6] The `qfilt` program allows students to understand the ramifications of issues such as coefficient quantization and realization tradeoffs, but also provides a bridge to real-time DSP. Note the button labelled Load/Run DSK (on the right, fifth button from the bottom). Clicking this button takes the filter coefficients the student has produced, loads an FIR filter routine on the DSK that implements this design, and runs the DSK in real-time to demonstrate the filtering effect. This gets the student past the initial fear of dealing with real-time DSP hardware. It is amazing how profound an effect can be observed in most students who listen for the first time to the sound produced by a real-time DSP filter that they designed themselves! After this experience, we move the student toward creating their own code in C for the DSK using the Code Composer Studio software tools which come with the DSK, shown in Figure 4.

Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition

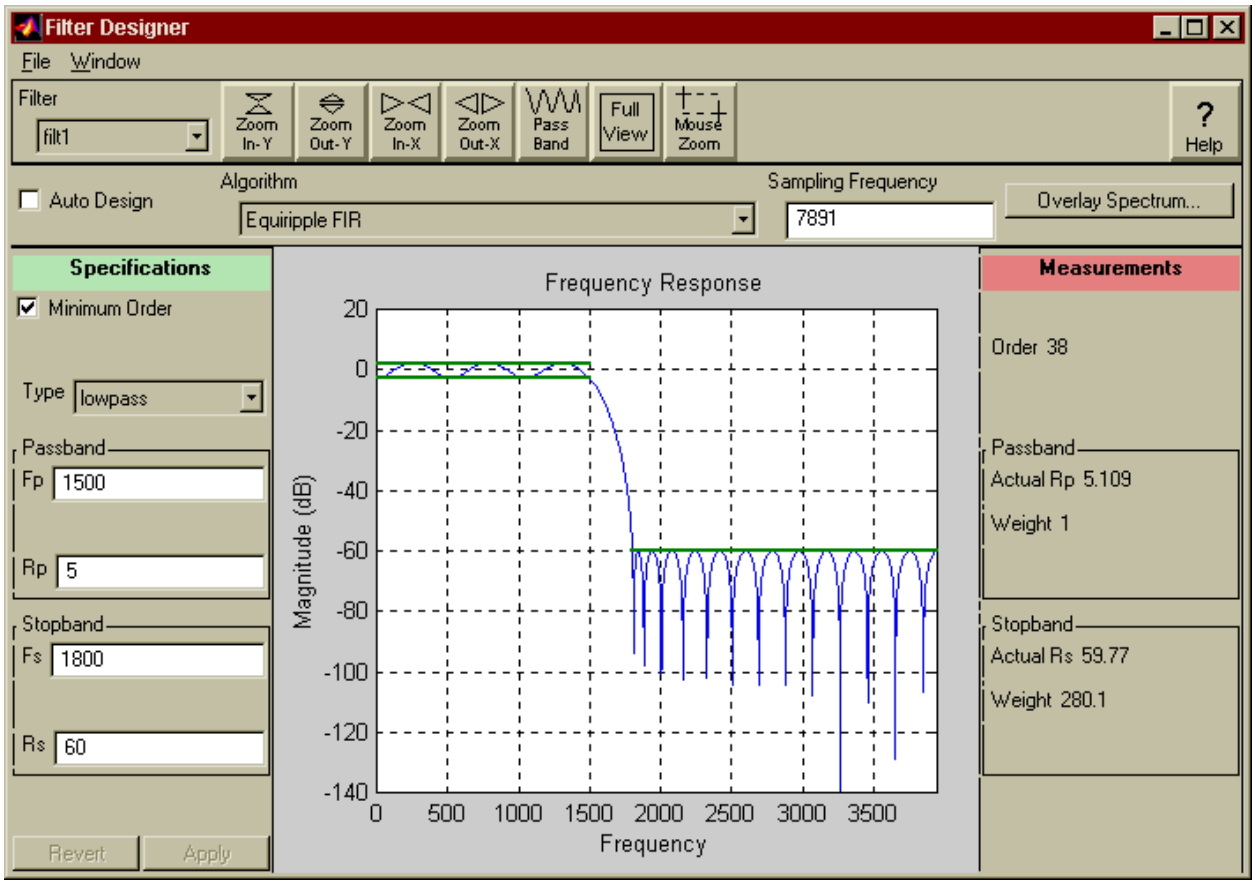

Figure 2: Example of the Graphical User Interface (GUI) of `sptool`.

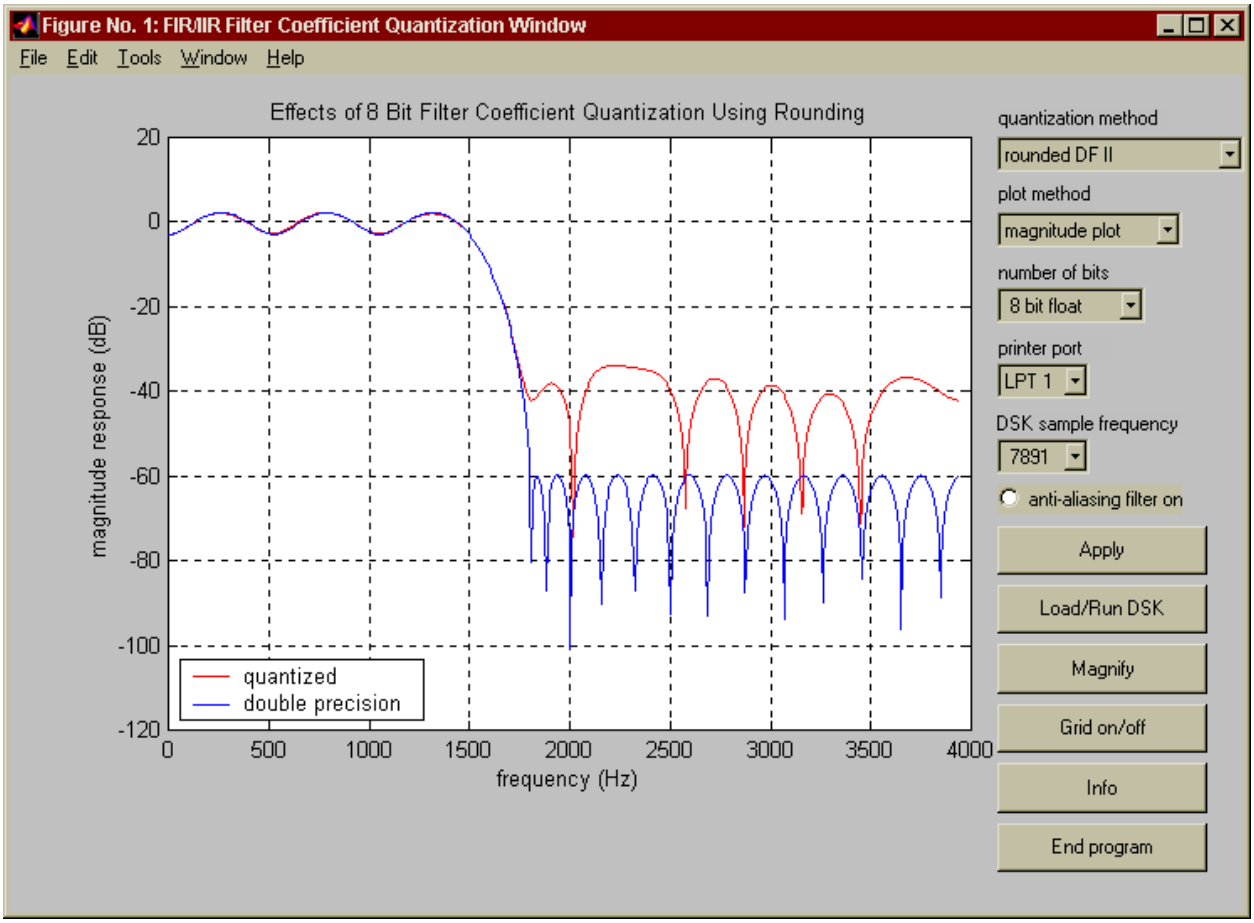

Figure 3: Example of the `qfilt` MATLAB program.

Eventually the student creates a working "brute force" FIR filter that runs on the DSK. We provide helper programs for MATLAB which format filter coefficients created in MATLAB into an "include" file that can be easily used by a C program. We encourage the student to further refine the C program, implementing, for example, circular buffering. Before long, the student is almost effortlessly validating filter designs in MATLAB, then moving them to the DSK using C.

3.2 Using the Hilbert Transformer

Once our students have mastered the basic implementation issues associated with FIR filters, a real-world application is needed. For our example here, we have selected a DSP-based envelope detector for a communications system receiver. This type of detector can be used to recover the message associated with a commercial amplitude modulation (AM) signal. A brief review of the theory follows.

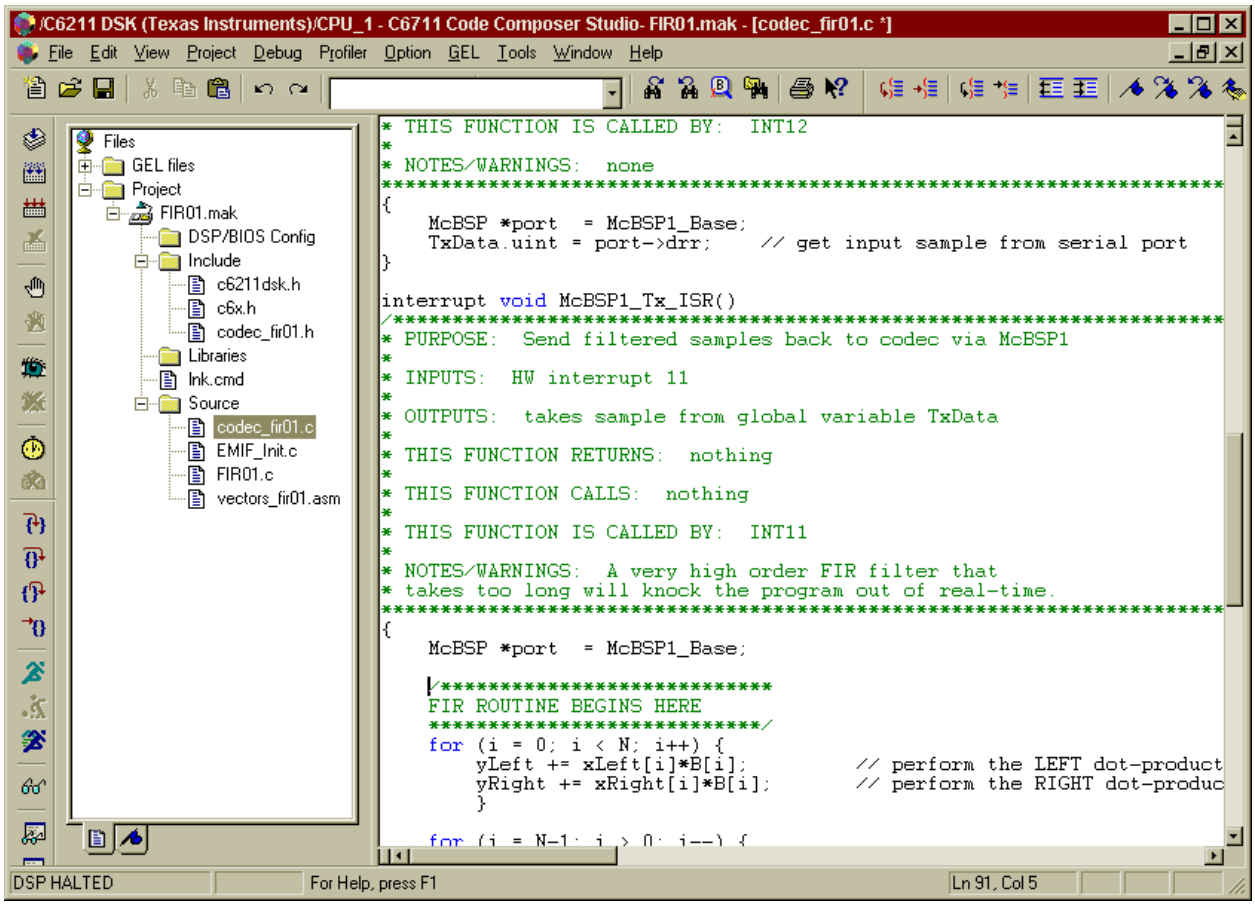

Figure 4: Example of Code Composer Studio for the DSK.

The expression for a double-sideband (with carrier) AM signal is,

$$s_{\mathrm{AM}}(t) = A_c[1 + m(t)]\cos(\omega_c t) \tag{2}$$

In this equation, A_c is the amplitude of the carrier, $m(t)$ is the message signal (with amplitude ≤ 1 to prevent overmodulation), and ω_c is the carrier frequency expressed in radians/sec.[25] In order to recover the message signal, it is necessary to extract the envelope of the signal $A_c[1 + m(t)]$. Once the envelope is obtained, the DC component can be removed with a DC blocking filter, leaving $A_c m(t)$, which is a scaled version of the original message signal.

The general principle of message recovery using DSP techniques is to select only the positive (or negative) frequency component of the signal* and determine its magnitude, which will be proportional to the envelope.[26] A Hilbert transformer filter will generate an all-pass 90° phase-shifted version of the received signal that is called the Q (for "quadrature") component. The non-phase-shifted version of the received signal is called the I (for "in-phase") component. Note that the analytic signal $z(t)$, defined as

$$z(t) = I(t) + jQ(t) \tag{3}$$

contains only positive frequency components. An important learning step for our students is to realize that they must account for the group delay of this FIR filter in order to align the I component with the Q component of the AM signal. At this point in the receiver development, the envelope may now be expressed as

$$\text{envelope of } s_{\mathrm{AM}}(t) = \sqrt{I^2(t) + Q^2(t)} \tag{4}$$

which means the envelope can be extracted using DSP techniques. The square root operation in Equation 4 may be directly implemented (for example using the floating point `sqrtf` command available via the DSK's C compiler) or by using a less computationally intensive approximation technique. As our teaching model suggests, the student first learns the theory with the aid of interactive demos, then develops a working solution off-line in MATLAB, then eventually moves to the DSK and implements a fully functional real-time DSP solution.

Example plots from a student project are shown in Figures 5–7. The AM signal shown in Figure 5 is a 2 kHz carrier modulated by a 700 Hz sinusoidal message signal. Note: in an actual software radio this would be the IF output. Figure 6 shows the impulse response of the Type III FIR filter the student designed using MATLAB's `remez` program to implement a Hilbert transformer on the DSK. A comparison of the original message and the recovered message using this DSP technique is shown in Figure 7. Note how well this design works!

We observe significant excitement and enthusiasm of our students when they can see real-time DSP-based demodulation working as a result of their own design.

*Recall that a *real* sinusoidal signal made up of positive and negative frequency components can be thought of as two counterrotating vectors; a *complex* sinusoidal signal, sometimes called an analytic signal, is made up of only a positive or negative frequency component and can be thought of as a single rotating vector. Signals made up of multiple frequencies can be treated similarly.

Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition

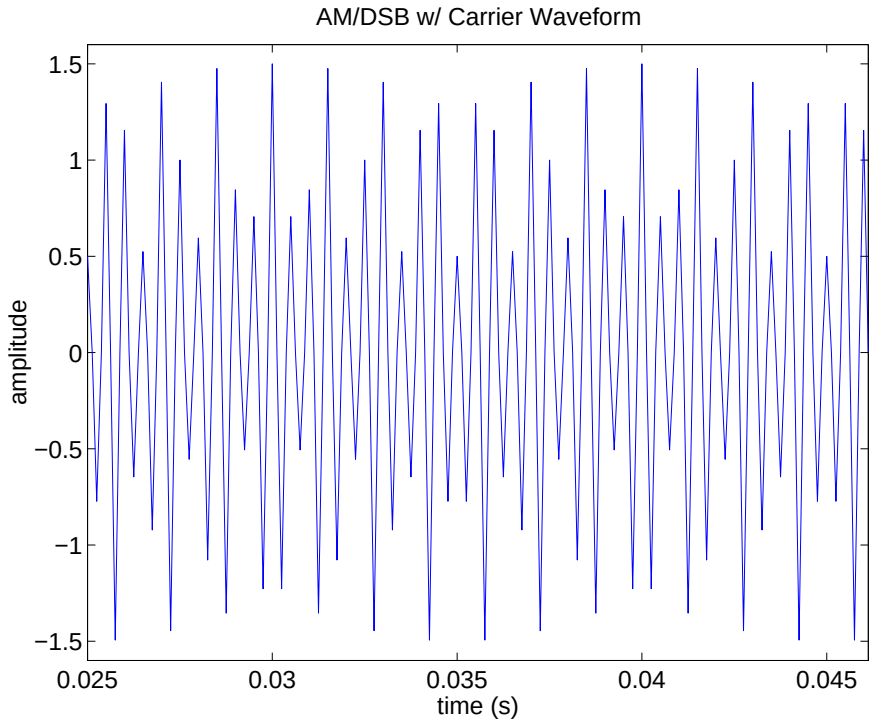

Figure 5: An AM signal to be demodulated via DSP.

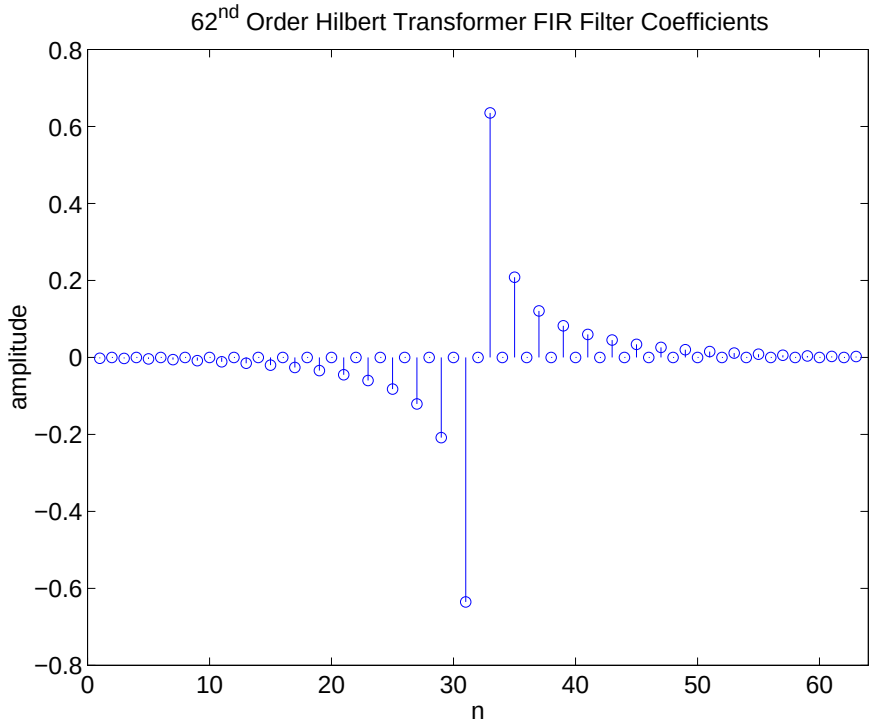

Figure 6: Impulse response of the Hilbert transformer Type III FIR filter used for the demodulation.

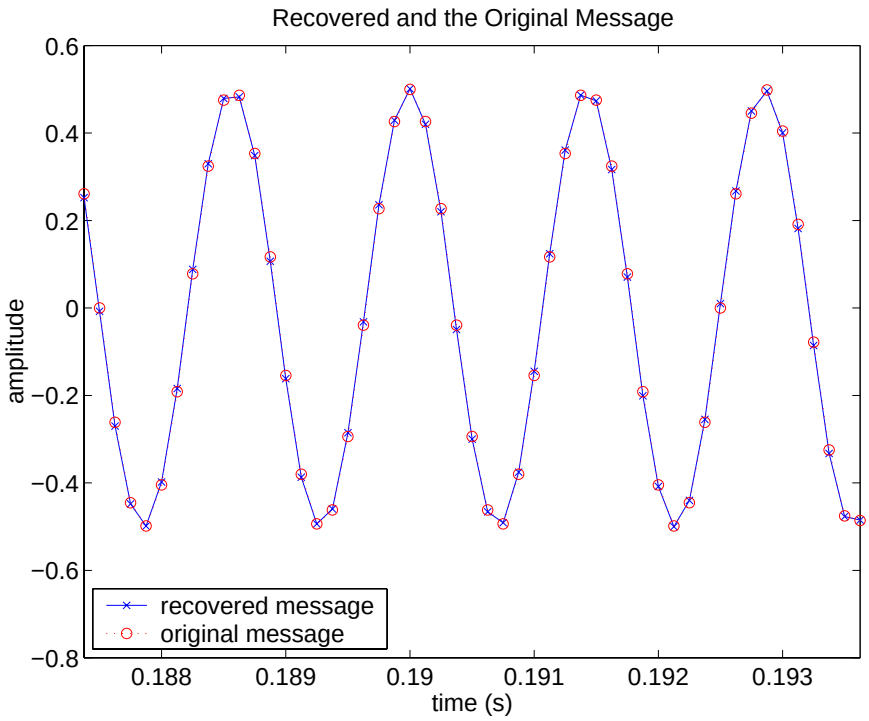

Figure 7: A comparison of the original message signal with the signal demodulated via DSP.

3.3 Audio Special Effects: Flanging and Chorus

An area of DSP in which many students are very interested is audio special effects. Altering the sound of an electric guitar or voice in real-time using student designed programs for the DSK has been extremely motivational for the students. We first describe to them the basic theory of special effects such as flanging and chorus, provide some demonstrations, then challenge the students to write algorithms in MATLAB using stored sound files.

A block diagram of the flanging effect is shown in Figure 8, where α is a scale factor, and $\beta[n]$ is a periodically varying delay described by

$$\beta[n] = \frac{R}{2}\left(1 - \cos\left(\omega_0 n\right)\right). \tag{5}$$

In Equation 5, R is the number of sample-time delays and ω_0 is a relatively low frequency.

A block diagram of the chorus effect is shown in Figure 9. To generate the chorus effect, three separately flanged signals are summed with the original signal. For a proper chorus effect, each of the β_i and α_i factors should be independent.

To ease the students on to the "bridge" that will get them to real-time processing, we first provide them with a custom program called `winDSK6` which provides a highly flexible graphical user interface that can easily manipulate the C6711 DSK (see Figure 10).

The `winDSK6` Audio Effects module contains a mixture of both FIR and IIR applications.

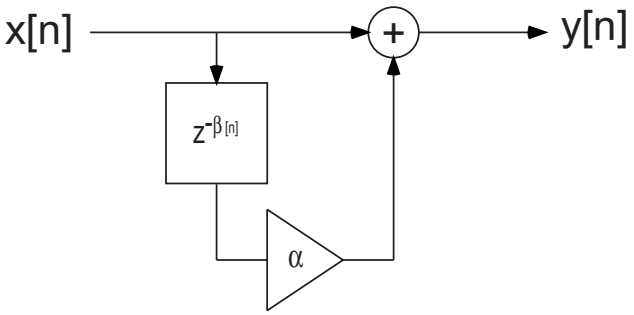

Figure 8: A block diagram of the flanging effect.

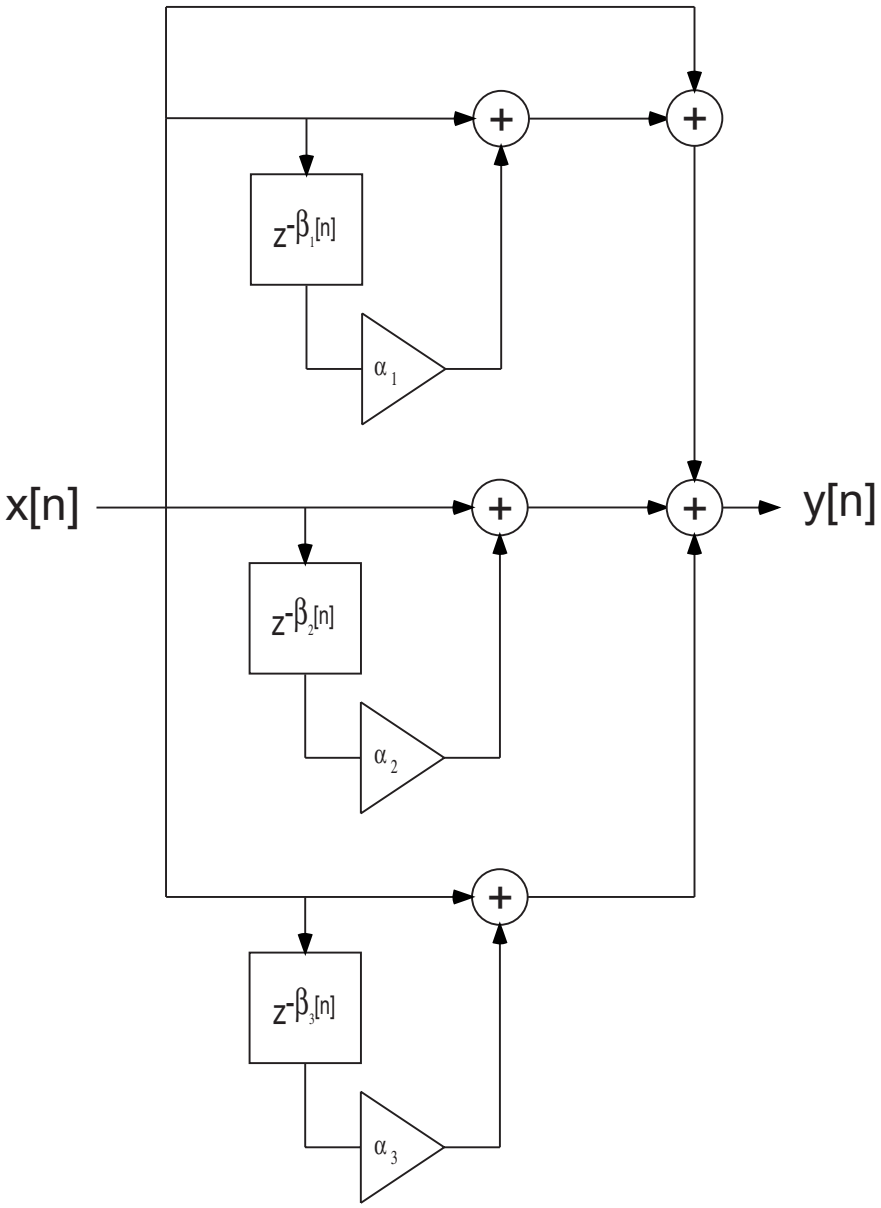

Figure 9: A block diagram of the chorus effect.

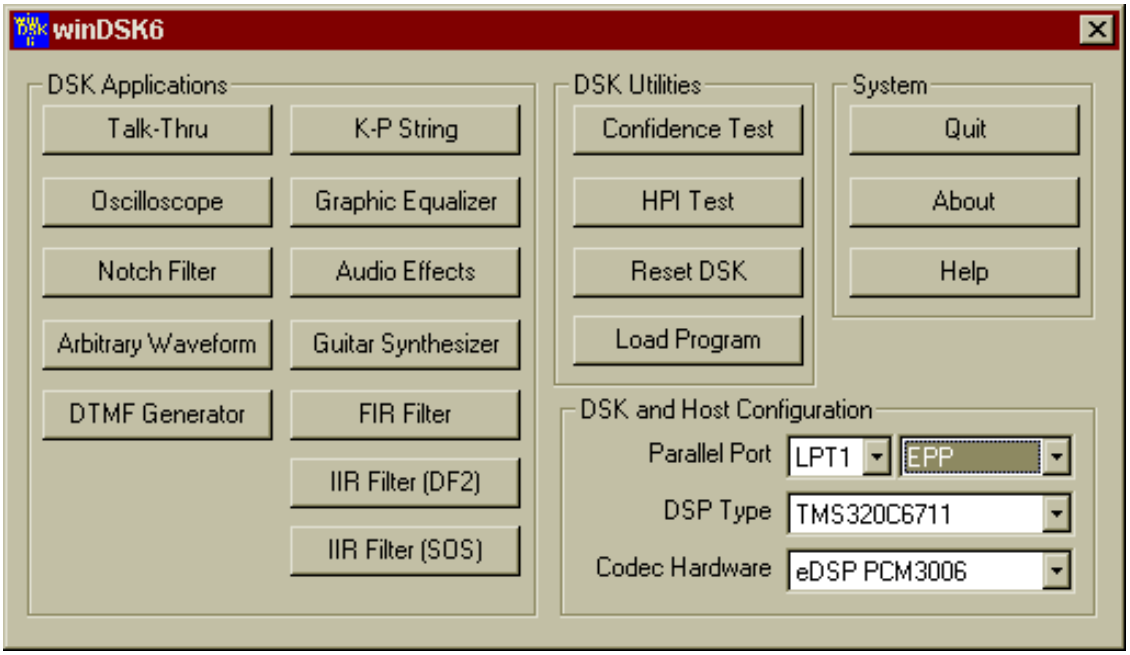

Figure 10: The `winDSK6` program's main window.

Clicking on the Audio Effects button from the `winDSK6` main window will load the audio effects program module into the attached DSK, and a window similar to Figure 11 will appear. The flanging and chorus effects are both implemented with FIR filters. After becoming comfortable with their own MATLAB programs and `winDSK6`, the next step is for them to write similar programs of their own in C that will run in real-time on the DSK.

Audio special effects have been particularly popular as capstone senior design projects, where the student designs and builds a unit which typically contains a C6711 DSK, power supply, interface buffers/amplifiers for microphones and/or electric guitars, and various user controls.

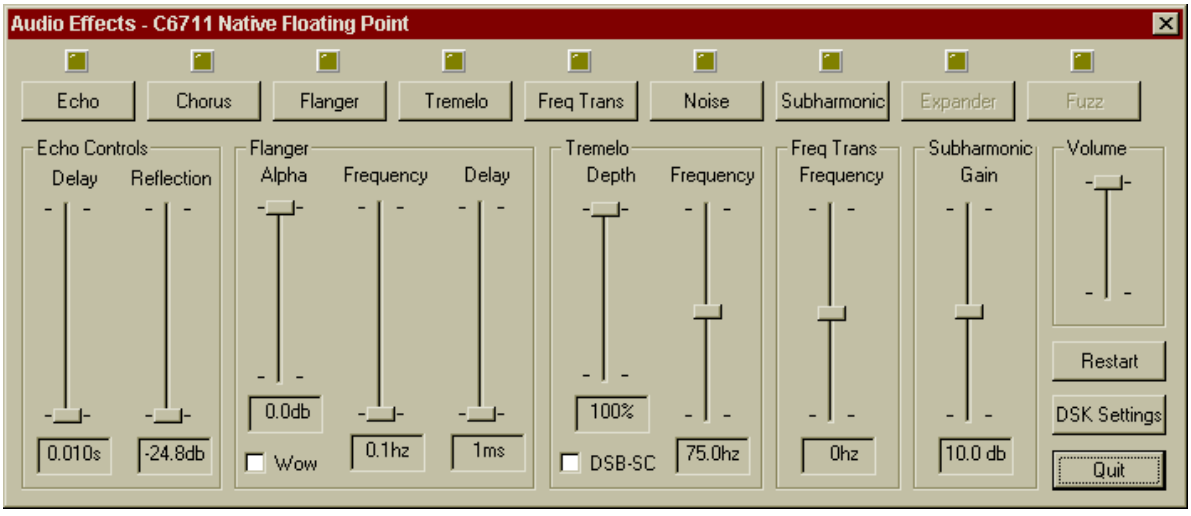

Figure 11: The `winDSK6` program running the Audio Effects application.

The unit produces whatever real-time audio special effect is created by the student's software, such as flanging, chorus, echo, reverb, harmonic generation, etc. This project effectively combines hardware and software in a single capstone design experience. It should be noted that until we implemented our "bridging the gap" method of teaching real-time DSP we could not entice a single student to pursue such a senior design project; after implementing the method there have been multiple students each year who enthusiastically chose such a project.

4 Conclusions

We have developed a systematic teaching model that allows our students to firmly grasp new DSP topics, and to smoothly transition from theory to a real-time DSP system implementation. Experiences at our respective institutions have shown that, compared to traditional undergraduate DSP classes, this model promotes greater comprehension of the theoretical underpinnings as well as the practical ramifications of each new DSP topic. The hardware investment required to implement such a model is rather modest, and much of the software needed has already been developed by the authors.

We freely distribute much of the associated software for educational, non-profit use, and invite user suggestions for improvement. See reference [27] for downloading the software; any interested parties are also invited to contact the authors via e-mail.[†]

References

[1] R. F. Kubichek, "Using MATLAB in a speech and signal processing class," in *Proceedings of the 1994 ASEE Annual Conference*, pp. 1207–1210, June 1994.

[2] C. S. Burrus, "Teaching filter design using MATLAB," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 20–30, Apr. 1993.

[3] R. G. Jacquot, J. C. Hamann, J. W. Pierre, and R. F. Kubichek, "Teaching digital filter design using symbolic and numeric features of MATLAB," *ASEE Comput. Educ. J.*, vol. VII, pp. 8–11, January-March 1997.

[†]For those interested in a more in-depth treatment of this "bridging the gap" method, a new book is nearing completion. It will include a guided step-by-step mastery of real-time DSP concepts using the TI C6711 DSK, many detailed laboratory experiments, all required background information on hardware and software issues for the C6711 DSK, and complete support software (MATLAB, C, and assembly). Please contact Dr. Welch for information on this book.

[4] C. H. G. Wright and T. B. Welch, "Teaching DSP concepts using MATLAB and the TMS320C5X," in *Proceedings of the 1998 Texas Instruments DSP Educators and Third-Party Conference*, (Houston, TX), August 6–8, 1998.

[5] C. H. G. Wright and T. B. Welch, "Teaching DSP concepts using MATLAB and the TMS320C31 DSK," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Mar. 1999. Paper 1778.

[6] C. H. G. Wright and T. B. Welch, "Teaching real-world DSP using MATLAB," *ASEE Comput. Educ. J.*, vol. IX, pp. 1–5, Jan–Mar 1999.

[7] M. G. Morrow, T. B. Welch, and C. H. G. Wright, "An inexpensive software tool for teaching real-time DSP," in *Proceedings of the 1st IEEE DSP in Education Workshop*, (Hunt, TX), IEEE Signal Processing Society, Oct. 2000.

[8] T. B. Welch, C. H. G. Wright, and M. G. Morrow, "Poles and zeroes and MATLAB, oh my!," *ASEE Comput. Educ. J.*, vol. X, pp. 70–72, Apr. 2000.

[9] M. G. Morrow, T. B. Welch, C. H. G. Wright, and G. W. P. York, "Demonstration platform for real-time beamforming," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001. Paper 1146.

[10] M. G. Morrow, T. B. Welch, and C. H. Wright, "An introduction to hardware-based DSP using winDSK6," in *Proceedings of the 2001 ASEE Annual Conference*, (Albuquerque, NM), June 2001. Session 1320.

[11] G. W. York, M. G. Morrow, T. B. Welch, and C. H. Wright, "Teaching real-time sonar with the C6711 DSK and MATLAB," in *Proceedings of the 2001 ASEE Annual Conference*, (Albuquerque, NM), June 2001. Session 1320.

[12] V. K. Ingle and J. G. Proakis, *Digital Signal Processing Using* MATLAB *V.4*. Bookware Companion Series, PWS Publishing, 1997.

[13] B. Porat, *A Course in Digital Signal Processing*. John Wiley & Sons, 1997.

[14] M. A. Yoder, J. H. McClellan, and R. W. Schafer, "Experiences in teaching DSP first in the ECE curriculum," in *Proceedings of the 1997 ASEE Annual Conference*, June 1997. Paper 1220-06.

[15] A. Ambardar and C. Borghesani, *Mastering DSP Concepts Using* MATLAB. Prentice Hall, 1998.

[16] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, 2nd ed., 2001.

[17] J. H. McClellan, C. S. Burrus, A. V. Oppenheim, T. W. Parks, R. W. Schafer, and S. W. Schuessler, *Computer-Based Exercises for Signal Processing Using* MATLAB 5. MATLAB Curriculum Series, Prentice Hall, 1998.

[18] The MathWorks, Inc., Natick, MA, *MATLAB: The Language of Technical Computing*, 2000.

[19] Texas Instruments, Inc., "C6711 DSK," 2001. http://focus.ti.com/docs/tool/toolfolder.jhtml?PartNumber=TMDS320006711%#devices.

[20] Texas Instruments, Inc., "Digital signal processing," 2001. http://dspvillage.ti.com/.

[21] C. H. Wright, T. B. Welch, and M. G. Morrow, "Teaching transfer functions with MATLAB and real-time DSP," in *Proceedings of the 2001 ASEE Annual Conference*, (Albuquerque, NM), June 2001. Session 1320.

[22] Educational DSP, L.L.C., "DSP resources for TI DSKs," 2002. http://www.educationaldsp.com/.

[23] C. Inacio and D. Ombres, "The DSP decision: Fixed point or floating?," *IEEE Spectrum*, pp. 72–74, Sept. 1996.

[24] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Prentice Hall, 2nd ed., 1999.

[25] L. W. Couch, II, *Digital and Analog Communication Systems*. Prentice Hall, 6th ed., 2001.

[26] M. E. Frerking, *Digital Signal Processing in Communication Systems*. Van Nostrand Reinhold, 1994. 7th printing 2000 by Kluwer Academic Publishers.

[27] M. G. Morrow, "University of Wisconsin at Madison," 2002. http://eceserv0.ece.wisc.edu/~morrow/software/.

CAMERON H. G. WRIGHT, Ph.D, P.E., is a Professor and Deputy Department Head of the Department of Electrical Engineering at the U.S. Air Force Academy, Colorado Springs, CO. His research interests include signal and image processing, biomedical instrumentation, communications systems, and laser/electro-optics applications. Lt. Colonel Wright is a member of ASEE, IEEE, SPIE, NSPE, Tau Beta Pi, and Eta Kappa Nu. E-mail: c.h.g.wright@ieee.org

THAD B. WELCH, Ph.D, P.E., is an Associate Professor in the Department of Electrical Engineering at the U.S. Naval Academy, Annapolis, MD (from 1994–1997 he was an Assistant Professor in the Department of Electrical Engineering at the U.S. Air Force Academy). His research interests include multicarrier communication systems analysis and signal processing. Commander Welch is a member of ASEE, IEEE, and Eta Kappa Nu. E-mail: t.b.welch@ieee.org

DELORES M. ETTER, Ph.D, is a Professor in the Department of Electrical Engineering at the U.S Naval Academy, Annapolis, MD, and holds the ONR Distinguished Chair in Science and Technology. From 1998–2001, she was the Deputy Under Secretary of Defense for Science and Technology. She is author of a number of engineering textbooks and her research interests include adaptive signal processing. Professor Etter is member of the National Academy of Engineering, a Fellow of the IEEE and the ASEE, and a member of Tau Beta Pi and Eta Kappa Nu. E-mail: etter@usna.edu

MICHAEL G. MORROW, P.E., is a Faculty Associate in the Department of Electrical Engineering at the University of Wisconsin, Madison, WI (from 1996–2000 he was a Master Instructor in the Department of Electrical Engineering at the U.S. Naval Academy). His research interests include real-time digital systems, power system automation, and software engineering. He is a member of IEEE. E-mail: morrow@ieee.org