AC 2007-1924: FIXED-POINT DSP IMPLEMENTATION: ADVANCED SIGNAL PROCESSING TOPICS AND CONCEPTUAL LEARNING

Wayne Padgett, Rose-Hulman Institute of Technology
Wayne T. Padgett received his Ph.D. from Georgia Institute of Technology in 1994. He has been teaching digital signal processing and related courses at Rose-Hulman Institute of Technology for 12 years. He is a member of ASEE, a senior member of the IEEE, and is on the IEEE Signal Processing Society's Technical Committee on Signal Processing Education.

Fixed-Point DSP Implementation: Advanced Signal Processing Topics and Conceptual Learning

Abstract

In this paper a description of a unique fixed point systems course, including a list of topics, a description of labs, and a discussion of the focus on a course project. The course has run four times using simulation environments to promote analysis and visualization. The content of the course has made it apparent that there are numerous linkages to advanced signal processing topics, and these are described. The course has also led to the initiation of an educational experiment using the Signals and Systems Concept Inventory (SSCI) to measure how two very different electives affect student understanding of basic concepts. The experiment compares the fixed point course which is very lab oriented, to a theoretical elective. Preliminary results are described. Work to develop a course text and lab materials is described as part of an effort to promote the adoption of fixed point material widely in electrical and computer engineering curricula.

Introduction

Fixed-point implementation issues in digital signal processing (DSP) are not widely taught or deeply covered in most U.S. undergraduate (or graduate) curricula. There seems to be a perception among faculty that fixed-point implementation is difficult to tie to theory, and not important for advanced work in the field. However, the author's experiences show that industrial practitioners rely heavily on fixed-point implementation skills, and that many opportunities exist to link a fixed-point implementation course to advanced signal processing topics. The author has taught a fixed-point system design course four times, and each time the need to draw on advanced topics has become more obvious.

Course Description

The course is called DSP System Design, not Fixed-Point Algorithm Development, because the only way to give students adequate experience with the tradeoffs and performance issues involved is to build the course around a project. Learning to measure, specify, and adjust the system's performance is a critical element of the course, and it drives the students to deepen their understanding of the fixed-point effects. Although many possible projects could serve well, the course has been based on an SSB communication system which takes input speech at an 8 kHz sample rate and then raises the sample rate to 96 kHz (12x) for SSB modulation at 40 kHz (actually four channels are implemented eventually). The receiver removes out of band interference, demodulates the signal, and then reduces the sample rate back to 8 kHz. The basic design is then extended to operate at four channel frequencies with minimal inter-channel interference, and subject to various (conflicting) performance criteria such as speech quality, SNR, and computational complexity.

In the past three offerings, all of the course project work has been done in MATLAB, primarily using the Fixed Point Toolbox functions. All of the fixed-point system development is done as

simulation in MATLAB with no hardware-specific restrictions other than the assumptions of 16 bit registers and 32 bit accumulators. It would be nice to extend the implementation further into a hardware platform, but this material is yet to be developed, and may prove to be too much content for a ten week term.

When the course was originally conceived, the author reviewed many texts[1] and finally used Ifeachor[2] in the first offering because of its emphasis on implementation, but because the students and the instructor use Oppenheim[3] in the graduate DSP course (ECE580), it was usually easier to reference review material in Oppenheim rather than Ifeachor. As a result, Oppenheim has been used in subsequent offerings, which is popular with students since they need only one text for both courses. However, Oppenheim is not an ideal text for the course either, because it is not intended as a text on implementation, and it does not offer support for the system performance aspect of the course.

Weekly Topics

Rose-Hulman operates on ten week terms. The main topics of the course are shown below for each week of the term. The ordering of the topics is intended to deliver the necessary information "just in time" to allow the students to progress in their implementation of the labs leading up to the SSB system project.

- Introduction and filter structure review
- Binary arithmetic, quantization noise, and overview of fixed-point effects
- Coefficient quantization, signal to noise ratio (SNR), roundoff noise, filtering random processes
- Overflow, scaling, examination of MATLAB's implementation of scaling for second order sections
- Single sideband modulation, Hilbert transform, single sideband receiver with interference
- Implementation of Hilbert transform (1 filter + delay, 2 filters, infinite impulse response (IIR)), upsampling and downsampling
- Polyphase, staged interpolation
- Sinsusoid generation (table, oscillator)
- Estimating power spectral density, hardware implementation issues (processor specialization, field programmable gate array (FPGA) vs processor, architecture vs algorithm, cost, power, yield)
- Project time

"Just in time" delivery of the course topics is important to enhance the students' sense of the practicality of the material. Students tend to be motivated to take this course because they see it as useful in the "real world," and their positive impression is strengthened by immediately putting each topic to use in lab.

Labs

The lab topics are also focused on building up the skills and understanding necessary to succeed in doing the project. Because the first lab session normally occurs after just two days of class,

and because the students are often unfamiliar with the debugging and object-oriented features in LabVIEW, the first lab is intended to bring all the students to a minimum level of competency in using the debugging features, and in manipulating the filter and fixed-point tools used in LabVIEW. The next six labs seek to match simulations with actual fixed-point filters and signals with the theoretical material covered in class. The first seven labs are done individually, to ensure that all students become competent with the Fixed Point Toolbox. The later labs are done in groups of two.

In each of the theory comparison labs, care must be taken to isolate the effects of a particular source of error from other types of error so that accurate comparisons with the theory can be made. For example, when measuring roundoff noise it is important to use quantized coefficients in the floating point reference system so that the effects of both roundoff noise and coefficient quantization are not combined.

Lab 8 is really a prologue to the project. The students are given a working implementation of an SSB system in floating point LabVIEW code, and they must convert to fixed-point, and make measurements of SNR and computational complexity. The actual project involves not just getting the system working, but optimizing according to performance criteria. The system is extended to operate at four different frequency channels, and measurements of inter-channel interference are included. These performance measures require the students to examine various implementations to see which perform best and which minimize computational cost. The critical element of the project is that it exposes students to real tradeoffs and constraints in a way that that is significant to them (their grade is effected). The final lab time is used for a wrap up session. The purpose of this session is to help all the groups benefit from the lessons learned by each group. Each team has a chance to share their intuition and see how their results compare in performance and hours logged from a comparison chart.

- Lab 1 – LabVIEW Tutorial
- Lab 2 – Second Order Sections
- Lab 3 – Quantization Noise
- Lab 4 – Integer Computation
- Lab 5 – Coefficient Quantization
- Lab 6 – Roundoff Noise
- Lab 7 – Scaling and Overflow
- Lab 8 – Fixed Point SSB
- Project – Optimized Multichannel SSB System (2 weeks)
- Wrap up Session

Advanced Topics

As noted above, the discussion of fixed point effects must address a number of the advanced topics surrounding signal processing. Several of these topics are not part of a typical signal processing course, and so the fixed-point emphasis becomes a link to multidisciplinary material. Fixed point implementation has been accused of being a collection of ad-hoc topics that are impossible to generalize, but the truth is that the difficult issues that arise are opportunities to expose students to topics in mathematical modeling, applications of random processes, etc.

Random Processes

Random processes are useful models for both signals and noise, and are important in communication systems courses as well as upper level DSP courses. In a fixed point implementation course, random processes are explicit in the approximate linear analysis of quantization noise, both in signal quantization and in the roundoff error introduced in most arithmetic. It is impossible to discuss roundoff error without some analysis of the filtering of random processes, usually not only determining output power but also power spectral density. The task of calculating SNR for fixed point systems (when the source of noise is quantization error) is critical to developing performance specifications, and it requires a clear understanding of filtering random processes.

Statistical Modeling

Even in the simplest demonstration of how quantization can be modeled as uniform distributed noise, the accuracy of the approximation becomes an issue. If the signal to be quantized is periodic, then the error may also be, resulting in a discrete distribution of errors. If the quantization levels are large, as may be the case for a "simple" demo, the error distribution may not be uniform. The distribution of quantization noise at the output of a filter is neither uniform nor Gaussian, but if we wish to estimate probabilities of overflow, we need an estimate of the cumulative density function. This exercise requires a substantial understanding of probability theory for an undergraduate, but it makes an excellent application to justify the need for these modeling techniques.

Filter Structures

In an introductory DSP course taught assuming ideal precision, introducing various filter structures can seem quite pointless to students. After all, they are all designed to achieve identical results. The reason for considering most alternative structures (speech modeling applications are an exception) is the need for minimizing coefficient quantization error, and sometimes overflow and roundoff error effects. It is no coincidence that these are the three main sources of fixed point error. The study of alternative filter structures is rather empty without including the analysis of the precision effects. In floating point, there is no reason to choose the increased complexity of an alternative structure, and in the minds of students, no reason to learn about them.

Coefficient Sensitivity Analysis

Students who discover the purpose of various alternative filter structures will eventually need to determine which structure best suits the needs of a particular application. Often it is possible to analyze a filter structure and determine via partial derivatives the sensitivity of the frequency response to variations in each coefficient. The predictive power of this technique (well known in analog electronics where component variation is the problem) can allow a designer to choose a filter structure with the maximum accuracy of a desired parameter at minimal coefficient precision.

A related issue is the relative accuracy requirements of different filter design methods. A filter design based on the Butterworth method may result in poles farther from the unit circle and less need for coefficient accuracy than a lower order filter based on an elliptic method. These effects are not too difficult to predict with some intuition about filter design methods and coefficient sensitivity.

Numerical Analysis

Students in signal processing courses are often completely unaware of the limitations of precision in most computer arithmetic. Because floating point calculations (such as the 64 bit "double" in LabVIEW) are treated as unquantized, it can be a shock to realize that errors are introduced in almost every calculation, and that if an algorithm is sufficiently sensitive to such errors, the results can be useless.

A simple example of the small errors introduced in a frequency domain convolution is enough to motivate the concept of "machine epsilon" for the arithmetic system in use, and show that the study of numerical sensitivity is a very important discipline with a strong relationship to signal processing. A related topic appears in the discussion of various rounding methods meant to prevent the accumulation of error in iterative algorithms.

Computer Architecture

The entire purpose of discussing fixed point systems is based on their relative advantages over floating point hardware. These advantages can be summarized in terms of cost and power. Both factors depend largely on silicon area, which is intimately tied to the architecture of the entire system. A major advantage of fixed point systems is that the data can usually be squeezed into 16 bit words (or less) instead of the 32 bit words required for floating point. The resulting savings in bus sizes and memory sizes has a dramatic impact on the system's cost and power.

Another key point that can be easily introduced in the discussion of fixed point algorithms is the need to optimize the hardware and the algorithm jointly. This becomes painfully obvious when analysis calls for a 17 bit quantity on a processor that only supports 16 bit data. Students with a thorough knowledge of fixed point effects have some hope of being able to adjust the algorithm to reduce the precision required in one quantity, perhaps at the expense of others where extra precision is available.

Phase Noise

The communication system at the center of the course described above requires a local oscillator. A significant portion of the course is devoted to various methods of generating sinusoidal signals efficiently, and the types of noise produced by each. Since the noise in the oscillator signal is a significant performance limitation for the system, students need to understand the sources of frequency error, amplitude noise, and phase noise, so that they can evaluate the advantages and disadvantages of digital oscillators and lookup tables (numerically controlled oscillators). Almost

all of these noise sources are due to quantization effects and could not easily be discussed in a course without a fixed-point component.

Polyphase Implementation

Polyphase implementation is not a strictly fixed point topic, but it is unavoidable when making comparisons of system computational cost as a performance specification. The system used for the course project is a multirate system, and therefore polyphase filter implementation not only reduces the computational cost of the system, but by allowing FIR filters to be practical, it reduces the scaling problems associated with the high order IIR filters that are replaced with polyphase filters.

Simulation

Most fixed point implementation material is tightly focused on specific hardware – either a particular processor or a particular FPGA. This situation has led to the impression that fixed point issues can only be taught at either a very shallow level, or with very specific hardware. In an academia, we prefer to teach material that is easy to generalize, and this impression has led many to avoid fixed point topics entirely. However, with good simulation tools, there is no need to give up a general view of the issues even if a specific hardware example is used for motivation. Both MATLAB and LabVIEW have excellent fixed point simulation and analysis tools available, and both have products designed to target processors and FPGAs in an automated fashion. The course described above has been run using both MATLAB and LabVIEW tools for simulation, and only minimal hardware implementation. Students have gained the sense of a realistic implementation, and the lecture material has remained quite general although focused on processor issues more than FPGA implementation issues. Of course, there are both advantages and disadvantages to depending almost entirely on a simulation environment in this course.

Pros

A primary benefit of simulation tools and their development environment is the presence of powerful analysis, visualization, and debugging tools. In a simulation environment, it is not difficult to log overflows, or collect statistics on data, or even to plot histograms of error values, and compare the fixed point output to floating point results. In most cases, the floating point results can be treated as "ideal" and so signal can be separated from quantization and other errors. This approach allows the validation of the theoretical predictions discussed in class.

If we wish to explore the effects of different configurations of precision such as reduced word lengths for FPGA simulation, or increasing the precision of a portion of the algorithm to 32 bit data, the simulation tools can easily be reconfigured to reflect the changes, and all the analysis tools still apply.

The use of simulation tools teaches good habits in system development. Students can develop and debug their algorithms in floating point, then simulate the results in fixed point and resolve any precision issues before starting any hardware implementation.

Cons

There are two major problems with using a simulation environment. The first is that students have a strong interest in practical applications, and restricting them to a simulation environment makes the course seem less "real," and therefore less motivational. The second is that simulating fixed point in a general framework requires a great deal of overhead so that simulations are far from real time and in fact require some strategies for minimizing testing time. Here, hardware platforms have a distinct advantage, in that they can implement the system at high speed, and even make real time demos possible, even if their debugging and analysis capabilities are much weaker.

Ideally, a fixed point course could cover the entire development process, starting with simulation and then after the project had been successfully designed and tested, moving on to a specific hardware implementation. Perhaps a comparison of processors and FPGAs could be implemented by having parts of the class use different implementation platforms, or even partition the design between the two. Unfortunately, such a complete version of the material has not yet been offered, and may not be practical in a ten week term.

Conceptual Learning Experiment

In the process of developing this course, the question was posed, "how will it affect student understanding of fundamental concepts?" The question of how this hands-on course would compare with a more theoretical course was also raised. Fortunately, a tool exists for measuring student understanding of fundamental signal processing concepts. It is called the Signals and Systems Concept Inventory (SSCI){cite} and it is available both in a continuous time (CT) and discrete time (DT) version. Since this test was already in use in the prerequisite DSP course, it seemed convenient to begin an experiment to determine the comparative effects of two electives on SSCI scores.

The two courses involved are ECE497 DSP System Design, and ECE580 Digital Signal Processing. Although ECE580 is a graduate level course, it is populated by many of the same students who take ECE497. The concept inventory uses a pretest and posttest model to measure how much students have improved during the course. This experiment is apparently the first use of the SSCI to examine elective effects on student learning. Because posttest data is available from the introductory (and prerequisite) course ECE380 Intro to DSP, it is possible to see that students begin to lose familiarity with the material by the time they take the pretest in an elective course.
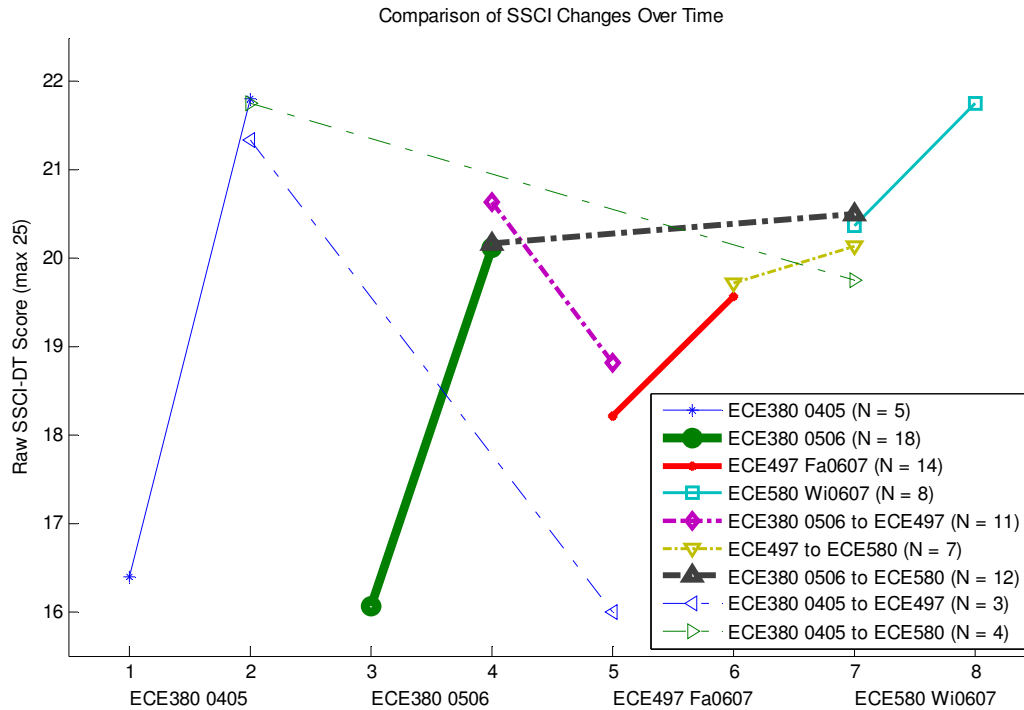
Fig. 1 – Average SSCI scores usually improve during a course and decline in between courses.

Fig. 1 shows how the average SSCI scores varied over time. Average scores for students who took two tests consecutively are shown. Since the students took the three courses in many combinations, the number of students in each group varies, and is shown in the legend as N. The line weights also reflect the value of N.

Some of the results in Fig. 1 are just as expected, such as, SSCI scores always increase during a course; scores increase more during the signals and systems course than during the electives; and scores tend to decrease in between courses. It is somewhat surprising that two groups of students increased their scores between courses: those taking ECE580 immediately after finishing ECE497, and those taking ECE580 after completing ECE380 the previous year. It seems significant that the group taking ECE497 starts and finishes with lower scores than the group taking ECE380 even though several took both courses (see Fig. 2 for more detail). The gain computation described below will magnify the difference between the ECE497 and the ECE580 groups even though their absolute change in scores is very similar.

It is only possible to speculate about why students may have increased their scores between courses. Some possible explanations might include reduced stress during the pre-test since the post-test is part of the final exam, or material learned in other courses during an intervening term. Another explanation could be greater familiarity with the SSCI test questions, since the pre-test in ECE580 would be the fifth attempt for students who took ECE497.

Wage uses a gain computation to express how much the students learned as a percentage of what they did not already know. Here, the gain is computed as

$$gain = \frac{post - pre}{25 - pre}$$

where 25 is the maximum test score, and average scores are used for each pre and post test. In discussions of this data with Wage, the need for a useful formula reflecting the loss became apparent, since gain is difficult to interpret when negative. For the purposes of this experiment, loss is defined as

$$loss = \frac{later - earlier}{earlier}$$

where the resulting negative fraction can be interpreted as how much of what the students once knew that has been forgotten. The terms "later" and "earlier" are used because the later test is the pretest, and the earlier test is the posttest in cases where loss occurs between courses.

Table 1 shows the gains calculated for the successive tests shown in Fig. 1. Positive gains are recorded during the courses, and a loss is shown for the periods where scores dropped. Although it is clear that the ECE580 students finished with a higher average than the ECE497 students, the sample size is small in this preliminary experiment, and a data collection error resulted in only volunteers taking the ECE580 post-test which may also have biased the results somewhat. Fortunately, the ECE580 pre-test results are for the entire class, and the high value of their average suggests that stronger students took ECE580 than ECE497.

First	Second	N	1st average	2nd average	gain	loss
1	2	5	16.40	21.80	0.63	
3	4	18	16.06	20.11	0.453	
5	6	14	18.21	19.57	0.20	
7	8	8	20.38	21.75	0.30	
4	5	11	20.64	18.82		-0.09
6	7	7	19.71	20.14	0.08	
4	7	12	20.17	20.50	0.07	
2	5	3	21.33	16.00		-0.25
2	7	4	21.75	19.75		-0.09

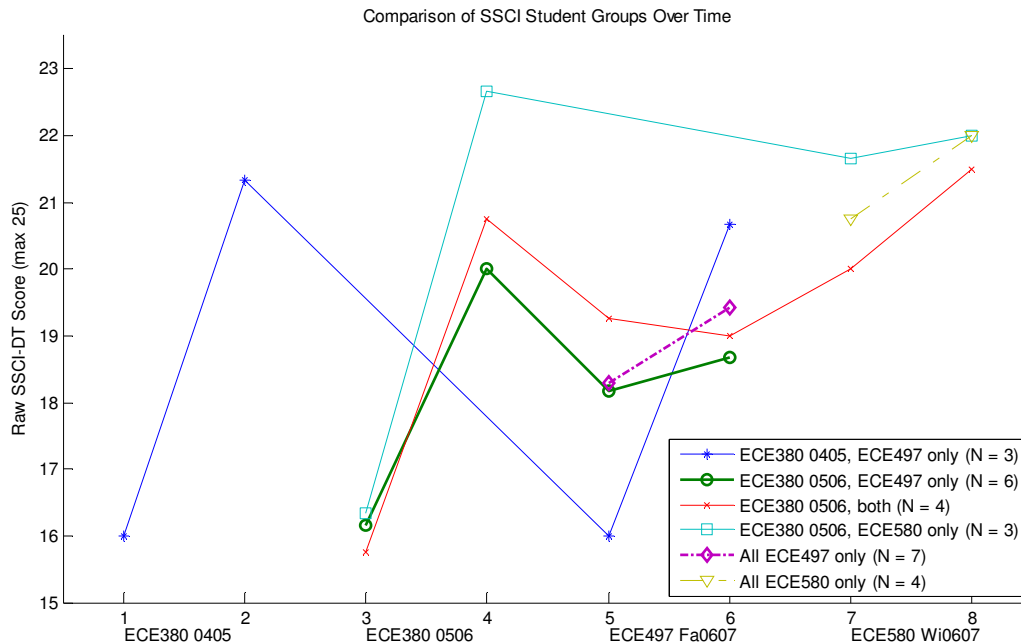Table 1 – Gains and losses in SSCI scores

Fig. 2 – SSCI scores grouped by students who took similar paths through the signal processing electives.

Fig. 2 shows the scores of similar groups of students varied over time. The first four lines shown in the legend are of students who took exactly the same sequence of tests, while the last ignore the year in which ECE380 was taken to create larger groups for comparison. In this figure, it is clear that the groups experienced very different results during ECE497. The group that took ECE380 during the 04/05 academic year seems to have lost all familiarity with signals during the intervening year, but recovered dramatically, while the group that took both ECE497 and ECE580 actually declined in SSCI score during ECE497 and then rebounded over the two-week quarter break, and finally exceeded their ECE380 average by the end of ECE580. The group that only took ECE580 was apparently very strong in ECE380, but never recovered their high average scores even after ECE580. These wide variations in results seem to suggest that the sample size is too small to be useful.

Future Work

A number of efforts are underway to enhance the effectiveness of the ECE497 course. A textbook is under development which will support the theoretical core of the fixed point material as well as incorporating a set of homework problems and example labs. The planned text outline will include material emphasizing both processor and FPGA implementation. The intent of the text is to allow faculty with minimal expertise in fixed point to offer this material in an elective course. Schools interested in using a draft of the text will be considered as "beta testers." As noted in the list of advance topics above, great potential exists to expand the planned textbook to include material on many related areas.

The dataset size for the SSCI elective experiment is too small to make significant generalizations, and schools interested in collecting this data over a larger population would be

welcome to participate. As data is collected over several years for multiple schools the results should be more useful and consistent.

Conclusions

Fixed point implementation is a topic that is drastically underserved in electrical and computer engineering programs across the U.S. It is a topic with an undeserved reputation as being too specific to individual hardware platforms. In reality, the core topics can be taught at a very general level using a system design context, and with or without reference to specific hardware. Industry demand is evident for this material since the vast majority of DSP processor and FPGA designs are implemented using fixed point arithmetic. As noted above, fixed point algorithm development is an excellent entry point to many advanced topics and deserves to be emphasized much more in undergraduate curricula than it currently is. The solution to this problem is a wider acceptance of the topic as a DSP elective, and the development of text and lab materials to support the resulting courses.

As educational research data is gathered on DSP electives, it will become more apparent not only how various electives reinforce basic concepts, but also which basic concepts are most used in upper level courses and therefore deserve greater emphasis.

Bibliography

1. Padgett, W.T., "An Undergraduate Fixed Point DSP Course," Proceedings of the 2nd IEEE Signal Processing Education Workshop, Oct. 2002, Pine Mountain, GA, pp. 302-305.
2. Ifeachor, E.C. and Jervis, B.W., *Digital Signal Processing*. A Practical Approach, Addison Wesley Publishing Company, 1995.
3. Oppenheim, A.V., Schafer, R.W., and Buck, J.R., *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.
4. Wage, K.E.; Buck, J.R.; Wright, C.H.G.; Welch, T.B., **"**The Signals and Systems Concept Inventory, IEEE Transactions on Education, vol. 48, no. 3, Aug. 2005, pp. 448 – 461.
5. Padgett, W.T., "Teaching Fixed-point Algorithm Development in a Systems Context," Proceedings of the 12th Digital Signal Processing Workshop and 4th Signal Processing Education Workshop, Sept. 2006, Jackson Hole, WY, pp. 297-301.