

**AC 2007-2711: TEACHING HARDWARE DESIGN OF FIXED-POINT DIGITAL
SIGNAL PROCESSING SYSTEMS**

David Anderson, Georgia Institute of Technology

Tyson Hall, Southern Adventist University

Teaching Hardware Design of Fixed-Point Digital Signal Processing Systems

David V. Anderson¹ and Tyson S. Hall²

¹Georgia Institute of Technology, Atlanta, GA 30332-0250, dva@ece.gatech.edu

²Southern Adventist University, Collegedale, TN 37315-0370, tyson@southern.edu

Abstract

Signal processing theory and practice are enabling and driving forces behind multimedia devices, communications systems, and even such diverse fields as automotive and medical systems. Over 90% of the signal processing systems on the market used fixed-point arithmetic because of the cost, power, and area savings that fixed-point systems provide. However, most colleges and universities do not teach or teach only a very little fixed-point signal processing. This issue is being addressed slowly around the country but now a new challenge or opportunity presents itself. As reconfigurable logic technology matures, field-programmable gate arrays (FPGAs) are increasingly used for signal processing systems. They have the advantage of tremendous throughput, great flexibility, and system integration. The challenge is that signal processing in FPGAs is a much less constrained problem than signal processing in special purpose microprocessors. The opportunity arises in that it is now possible to explore more options and, more especially, to take a more systems-level approach to signal processing systems. In short, designing a signal processing system using FPGAs provides opportunities to look at many system design issues and trade-offs in a classroom setting.

We have developed a course to teach signal processing in FPGAs at Georgia Institute of Technology and in this paper we consider the challenges and methods of teaching fixed-point system design in this course. We discuss the topics chosen and how they differ from traditional microprocessor-based courses. We also discuss how systems engineering concepts are woven into the course.

1 Motivation

This paper describes the early development of a system design curriculum that uses field programmable gate arrays (FPGAs) to implement digital signal processing (DSP) systems. The goals of this curriculum are several-fold, including:

- to prepare students for system design problems with which they are likely to be presented upon graduation;

- to teach students to transition “idealized” DSP experience into practical system design;
- to teach students how to design within the context of many competing performance metrics.

Over the past ten years, signal processing has made its way from a predominately graduate topic, to an undergraduate topic and often an introductory topic to electrical engineering. While there are likely many reasons for this, one compelling reason is that it is possible to teach many systems concepts using DSP. However, the focus is usually on theory and the labs are performed on computers using high precision floating-point arithmetic calculations. On the other hand, practical DSP systems (such as those in cell phones, DVD players, cameras, military systems, toys, etc.) tend to use limited precision fixed-point arithmetic and operate under stringent power, size, cost, and performance constraints. It is estimated that over 90% of the DSP microprocessors sold and virtually all FPGA-based systems use fixed-point arithmetic.

1.1 Finite Precision Signal Processing

Two possible types of numerical representations include *floating-point* and *fixed-point* formats. Floating-point number formats are similar to the scientific notation format of a calculator—they consist of a number (mantissa) and a scale factor (exponent). The primary advantage to using floating-point formats is that a very large range of numbers can be accurately expressed. Fixed-point number representations lack a scaling factor and so they can only express numbers within a limited range. The primary advantage to using fixed-point formats is that fixed-point operations are significantly more efficient than their floating-point counterparts. Therefore, fixed-point integrated circuits (ICs) are smaller (resulting in more ICs per wafer), consume less power, can be clocked faster, and are cheaper than equivalent floating-point ICs.

Numbers represented in digital systems are necessarily quantized, that is they are approximated by a finite set of numbers that can be represented digitally. The primary difficulties with fixed-point systems that must be addressed by designers are summarized in the following list:

Overflow: If a number is larger than can be expressed in the particular fixed-point representation, then overflow occurs. In this case, the representation may be undefined or it may saturate at the maximum amplitude that can be expressed.

Underflow: If a number is smaller than can be expressed in the particular fixed-point representation, then underflow occurs. For example, an 8-bit fixed-point number may represent the numbers from 0 to 255. To represent fractional values, the same 8-bit number could be scaled by 2^{-8} so that the values 0 to $0.9961 = \frac{255}{256}$ are available. Now the smallest non-zero number is $0.0039 = \frac{1}{256}$. In this case 0.0019 and any

smaller values would underflow and be expressed as 0, limiting the precision of the result. Overflow and underflow are complementary problems—a system designed to guarantee no overflow will have more underflow problems and/or increased quantization noise problems.

Signal Quantization Noise: This is related to the above two problems. Quantization noise is the difference between the actual number and the quantized representation. Quantization noise is often approximated as white noise added to a signal.

Filter Coefficient Quantization: When filter coefficients are quantized the characteristics of the filter may change dramatically. For example, a stable filter may become unstable.

Designing fixed-point systems is non-trivial for several reasons. First, quantization effects are non-linear and not easily understood analytically. Second, fixed-point design rules are often competing, making it difficult to find the best solution. Finally, each part of a system must be considered in terms of its own requirements and how the signal representations in that part affect all others. For these reasons, teaching fixed-point system design requires “hands-on” participation of the students.

1.2 FPGAs and Signal Processing

When designing custom hardware solutions, the fixed-point issues may be even more pronounced than for DSP microprocessor solutions. This is because it is often desirable to optimize speed and/or chip area by using different fixed-point word sizes in different parts of the systems or by using coarse approximations to filter coefficients to improve efficiency.

Unfortunately, students graduating from most DSP programs are left unequipped to deal with the challenges of DSP hardware design and hardware/software co-design. Many curricula include separate classes in both DSP theory and hardware modeling; however, there are few opportunities given to students to combine these two skills into a working knowledge of DSP hardware design [1, 2]. Students often struggle to bridge this gap between the theory and the hardware implementation of DSP systems [3]. This paper presents a pedagogical framework whereby students can leverage their previous knowledge of DSP theory and Very High-Speed Integrated Circuit Hardware Description Language (VHDL) hardware design techniques to design, simulate, synthesize, and test digital signal processing systems [4]. Additionally, a curriculum is described that uses this framework to teach DSP hardware design.

Many courses and textbooks have been developed for teaching real-time DSP concepts using DSP microprocessors [5, 6, 7, 8, 9, 10, 11, 12]. The program presented herein is not designed to replace or displace DSP microprocessor-based programs—the goals of the program are different. Programs that rely on DSP microprocessors as their primary implementation medium tend to emphasize software programming rather than hardware design. By using

Table 1: Real-time DSP Technologies and Their Pedagogical Usefulness

Function	FPGA	DSP μ P	MATLAB
Fixed-point Arithmetic/Operations	●	●	○
Floating-point Arithmetic/Operations	○	○/● ¹	●
Filter/Transform Implementations	○	●	●
Explore Design Trade-offs <i>(power, area, complexity, throughput, etc.)</i>	●	○	–
HW/SW Trade-offs	●	–	–
Parallel Processing	●	–	–
Data Flow/Buffering	●	●	–
Peripheral I/O	●	○	–
System Analysis	○ ²	○ ²	●

¹ There are both fixed-point and floating-point DSP microprocessors.

² System analysis/testing is possible with third-party MATLAB interfaces.

Key:

– = No or very limited support

○ = Possible

● = Efficient, well-suited to technology

field-programmable gate arrays (FPGAs) as the core technology, students are given the opportunity to design custom hardware implementations [13] and investigate concepts such as massively parallel algorithms. In addition, FPGAs can synthesize microprocessor cores allowing students to investigate the trade-offs between hardware and software implementations.

As illustrated in Table 1, FPGAs provide a very versatile platform for teaching real-time DSP implementations. While they are not optimal for every function, FPGAs do provide the most flexibility, which is valuable in design contexts. Students can use a single system to explore hardware and software designs and to make various design trade-offs to optimize their systems for power, area/size, complexity, throughput, latency, etc.

2 Previous FPGA DSP Courses

Georgia Tech has offered a course twice under the title ECE 4273 DSP Chip Design that attempts to bridge the gap between the theory and hardware implementation of digital signal processing systems [14, 4]. Many curricula include separate classes in both DSP theory and hardware modeling; however, there are few opportunities given to students to combine these two skills into a working knowledge of DSP hardware design [1, 2]. This course was designed to fill this void by allowing students to leverage their previous knowledge of DSP theory and very high-speed integrated circuit hardware description language (VHDL) hardware design techniques to design, simulate, synthesize, and test digital signal processing systems [4].

ECE 4273 Student Outcomes

Upon successful completion of ECE 4273, students should be able to:

1. Convert numbers between decimal, floating-point, and fixed-point number formats including Q-formatted numbers and canonical signed digits.
2. Synthesize digital logic and fixed-point signal processing systems using VHDL.
3. Design filters that are robust to quantization effects.
4. Design hardware filters using distributed arithmetic.
5. Optimize hardware filters given realistic design constraints using a variety of filter design techniques.
6. Design the hardware to implement an adaptive filter.
7. Describe the relevant theory and implementation of an adaptive filter.
8. Describe the trade-offs (including precision, accuracy, dynamic range, implementation size, and signal-to-noise ratio) between fixed-point and floating-point implementations.
9. Describe the advantages and disadvantages of implementing DSP systems in DSP microprocessors and dedicated hardware (FPGAs or ASICs).

Figure 1: Measurable student outcomes for ECE 4273.

ECE 4273 is a senior-level technical elective in digital signal processing. It is a three semester-hour course with two hours of lecture and a three-hour laboratory session each week. The lecture and laboratory material are closely coordinated such that topics covered in lecture are reinforced in a hands-on laboratory experiment within at most a two-week time period.

Undergraduate students taking this course are expected to be familiar with MATLAB, digital filter design, basic transforms, FPGAs, and VHDL. These assumptions are valid given the enforced prerequisites (a senior-level fundamentals of DSP course) and the required core curriculum for electrical and computer engineering majors at Georgia Tech, which includes laboratories and classroom lecture on FPGAs and VHDL in the digital design and computer architecture sequences [15, 16].

Since the nature of this course is a convergence of DSP and computer engineering (CMPE), lecture material is pulled from both of these disciplines. The course schedule typically consists of one week of lectures on DSP theory, optimization techniques, etc. followed by one week of implementation-related lectures. The laboratory projects then provide students with an opportunity to combine these two subjects into a working knowledge of DSP hardware design [17].

Students were assessed on the outcomes shown in Figure 1. First, laboratory projects throughout the semester provide ample opportunity for students to learn and demonstrate

their acuity at designing hardware systems (items 2, 4, and 5). In addition, the final design project in adaptive filter design requires students to demonstrate skills in items 2, 3, 4, 5, and 6. Finally, three exams are given including a comprehensive final exam that assesses students' achievement of items 1, 2, 3, 4, 7, 8, and 9.

2.1 DSP System Design

A separate, but related course, Real-time DSP System Design, was taught for the first time in 2005 at Georgia Tech. This course had a much smaller laboratory component and was designed to expose students to real-time DSP concepts and system design trade-offs.

3 Proposed FPGA DSP Curriculum

Having covered previous courses in DSP hardware design and system-level design, a refined fixed-point DSP hardware design curriculum that uses a system design approach will be presented. Teaching fixed-point hardware design is most natural within a hands-on, laboratory environment where real-world obstacles will be encountered and overcome. In the following subsections, the proposed objectives for a fixed-point DSP hardware design course are laid out. The objectives for the lecture and laboratory are presented separately to show the distinct aspects of each course component. Proposed assessment methods for evaluating these objectives are presented in the following section.

3.1 Laboratory Objectives

Feisel and Rosa have previously listed a rather exhaustive list of learning objectives for engineering instructional laboratories [18]. The objectives for the laboratory component of this proposed curriculum are shown in Table 2 along with their mappings to the Feisel objective categories.

The laboratory objectives for this curriculum are heavily oriented towards design, instrumentation, and data analysis. This cycle is repeated in laboratory projects throughout the semester. Early projects allow the students to explore basic implementation concepts such as buffering, fixed-point arithmetic, and quantization effects through the design and synthesis of low-order FIR filters. Later laboratory assignments cover subjects such as IIR filter design, filter optimization techniques, and multiplierless implementations (*e.g.*, distributed arithmetic). A specific schedule of laboratory assignments is given in Table 3. As shown in the table, each laboratory assignment targets at least one course objective.

Each laboratory has multiple components. A pre-lab exercise should be given to the students a week in advance, and it includes a problem analysis or theoretical study that ties the lectures to the laboratory assignment. When appropriate, students are also required to simulate the DSP module using engineering simulation software such as MATLAB or LabVIEW. During labs, students implement a hardware design in VHDL for their DSP system. Finally, the students test their design and analyze its performance relative to expected performance.

#	Curriculum Objectives	Fundamental Objectives
1	Demonstrate competence in the operation of software simulation tools and hardware design environments.	Psychomotor
2	Simulate quantization noise effects and theoretically predict quantization noise power spectra at the output of a fixed-point filter.	Models
3	Synthesize basic digital logic and basic DSP functions (buffering, table lookup, arithmetic, etc.) in an FPGA.	Design
4	Design filters that meet a given set of realistic system parameters.	Design, Creativity
5	Implement filters of various flavors and complexities that execute on an FPGA using distributed arithmetic.	Instrumentation, Data Analysis
6	Acquire experimental frequency responses for filters implemented in custom hardware. Compare these responses to software simulations of the same filters.	Instrumentation, Models, Data Analysis
7	In teams of two to four, design a DSP system that is optimally partitioned between hardware and software to meet a given set of realistic system parameters.	Design, Creativity, Teamwork
8	In teams of two to four, implement a DSP system that is optimally partitioned between hardware and software for a given set of constraints. Experimentally characterize the system to ensure that the given system parameters are satisfied.	Instrumentation, Experiment, Data Analysis, Teamwork

Table 2: Student Learning Objectives Mapped to Fundamental Objectives of Engineering Instructional Laboratories

The final laboratory project will be designed to be a four-week team design experience. Students should be given the choice of proposing their own design project or implementing the provided one. In either case, the final project will be required to include both a processor (software component) and custom DSP module (hardware component). These requirements allow the students to investigate hardware/software design trade-offs and attempt to achieve an optimal partitioning of their system. Students will be required to devise their own experiments to characterize their system and analyze their intermediate implementations to make

#	Laboratory Assignments	Course Objective #
1	Introduction to the Hardware Development Environment & Tools	1
2	HDL Design of Basic Digital Signal Processing Components	3
3	Simulation and Modeling of DSP Systems	1, 2
4	HDL Design of Buffers and FIR Filters	3, 5
5	Quantization Analysis of FIR Filters	6
6	Real-time FIR Filters and Qualitative Analysis	4, 5
7	Advanced Filtering with Third-Party IP Cores	4, 5, 6
8	Designing IIR Filters using Distributed Arithmetic	4, 5, 6
9	Implementing and Interfacing with Soft-Core Processors	1
10	DSP Hardware/Software Design Project	7, 8

Table 3: Laboratory Assignments Mapped to Laboratory Objectives From Table 2

improvements to their design. Since the final project is fairly open-ended, students must show original thought in their design and experimentation, which will lead to many creative projects and solutions.

The experimentation and data analysis aspects of this laboratory are important because they allow students to make connections between the physical hardware implementations and the DSP theory that they learn in lecture. Experimentally characterizing DSP systems can be a time-consuming task that requires very expensive laboratory equipment (lock-in amplifiers, spectrum analyzers, etc.). However, modern engineering software and current FPGA development kits can greatly simplify this process. In the past, the authors have developed a custom USB interface that allows students to create test signals, feed them through their physical hardware system, and acquire the output data all from within MATLAB [4, 19]. In addition, Mathworks' Simulink and National Instruments' LabVIEW both include the ability to program and interface with FPGAs from within their IDEs. With the appropriate lab equipment, these programs can greatly simplify/automate the process the acquiring system characterization data. These programs can also abstract the HDL coding aspects of these projects, if that is desired [20, 21].

3.2 Lecture Objectives

The objectives of the lecture component of this course are tightly coupled to the laboratory content. As shown in Table 4, the lecture objectives focus on the students' ability to explain theoretical aspects of fixed-point DSP systems, compare the advantages and disadvantages of different fixed-point system approaches, and calculate theoretical values for system parameters for example systems.

DSP hardware courses form a convergence of DSP and computer engineering (CMPE)

#	Course Objectives
1	Convert between decimal, floating-point, and fixed-point number formats including Q-format and canonical signed digits.
2	Explain the process of converting a floating-point system design to a fixed-point arithmetic implementation.
3	Describe the trade-offs (including precision, accuracy, dynamic range, implementation size, and signal-to-noise ratio) between fixed-point and floating-point implementations.
4	Use system performance criteria to compare two DSP system designs and determine which one is preferable.
5	Explain how different filter designs and structures have different effects on overflow, roundoff, and coefficient quantization.
6	Name and describe types of errors associated with fixed-point arithmetic and explain ways that these errors can be minimized.
7	Calculate noise power and power spectra at the input and output of an ideal digital filter.
8	Describe and discuss the advantages and disadvantages of implementations in a DSP microprocessor, FPGA, and ASIC.
9	Describe how various DSP programming techniques work including circular buffering, pipelining, interrupt processing, block processing, table lookup, and digital oscillator sinusoid generation.

Table 4: Student Learning Objectives for the Lecture Component

content. To support this convergence, lecture material must be pulled from both of these disciplines. The proposed schedule for this course follows a pattern of one week of lectures on DSP theory, optimization techniques, etc. followed by one week of implementation-related lectures.

DSP Material Covered The early lectures on DSP are designed to introduce students to the basic concepts needed in the rest of the course. Subjects to be covered include an introduction to real-time DSP systems, a discussion and comparison of different number formats, and an in-depth look at fixed-point arithmetic and the effects of quantization. The lectures during mid-semester focus on filtering as one of the core DSP building blocks found in signal processing systems. These topics include finite-impulse response (FIR) filter structures, filter design optimizations, the canonical signed digit format, sums-of-powers-of-two optimizations, subexpression sharing, and infinite-impulse response (IIR) filters. Toward the end of the semester, topics will be varied to emphasize the material needed for the final laboratory project. For example, floating-point number formats, adaptive filtering,

DSP microcontroller architectures, 2-D filtering, and hardware/software co-design are all plausible topics that can be covered in the final few weeks depending on the specific design project being assigned.

CMPE Material Covered It is assumed that a DSP hardware design course will be dominated by students specializing in DSP. Since DSP students may not be as familiar with implementation-related subjects, lectures on FPGAs, VHDL, and hardware implementations will start from a very basic point and progress quickly to the DSP implementation issues that this course should address. Thus, the early lectures focus on introducing and refreshing students on these subjects. Topics covered include basic FPGA concepts and architectures, introduction to VHDL for synthesis, and a review of state machine design with an emphasis on VHDL implementations. Lectures during mid-semester correspond to the DSP material being covered. Subjects covered include FIR filter implementations, hardware versus software fixed-point multipliers, and distributed arithmetic architectures. The semester ends with lectures on soft-core microprocessors, hardware implementation of a floating-point arithmetic unit, and implementations of basic adaptive-filter structures. In addition, commercial DSP microprocessor architectures will be investigated and used in discussions of implementation costs and trade-offs.

4 Assessment

The overall goal of the course is to teach students practical system design, particularly as it relates to signal processing. The specific learning objectives are accomplished via the laboratory projects and the coursework. These learning objectives are outlined in Tables 2 and 4. Table 3 provides a list of possible laboratory assignments along with associated objectives. Student assessment is designed to specifically measure these objectives.

4.1 Lecture Assessment

Much of the lecture is in preparation for the laboratory exercises, so in practice, student laboratory performance reflects student performance relative to the lecture material. However, lecture material is assessed independently using the following methods.

Homework Assignments

Fixed-point signal processing involves many concepts that students can practice through homework assignments. These include converting number formats including Q-format, evaluating filter scaling factors and stability, analyzing through-put and latency for specific structures, and designing and converting various structures such as distributed arithmetic filters.

Pre-lab Assignments

The pre-lab assignments directly assess and promote those parts of the lecture that tie into the laboratory assignments. For example, the students may be asked to design a

filter given some specific constraints. Their design will then be evaluated for practicality, and they will implement that design as part of the lab. The pre-lab assignments are very important for tying the lectures to the labs and also for assessing how well the students are able to apply what they have learned in lectures.

Quizzes and Exams

Quizzes are designed to test proficiency in basic signal processing and computing operations such as number formats, filter structures, and implementation syntax (*e.g.* VHDL code or MATLAB code). Quizzes also assess students' understanding of general concepts. Example questions along this line:

- What factors can cause filter coefficient quantization to more pronounced or severe?
- What are the effects of signal quantization / rounding on the filter output?
- Why/when are multi-rate filters used in signal processing applications?

4.2 Laboratory Assessment

Laboratory performance is the principal metric by which the student is judged to have met the course objectives. Student performance is assessed in terms of individual performance and, for the final project, individual performance within a team. The labs are specifically designed to meet the objectives of the course so that the objectives from [18] listed in Table 2 are a natural outcome of adequate lab performance. Each laboratory assignment is evaluated according to six dimensions that are closely related to the fundamental objectives discussed in the previous section: tools & environment, design, implementation, analysis, creativity, and teamwork. The grading rubric illustrated in Table 5 includes these six dimensions and is used for each lab assignment. By using a consistent rubric across lab assignments, the students' level of learning for each course objective can be monitored throughout the semester and averaged across the class. Students' individual lab assignment grades are calculated as a weighted average of the six dimensions on the rubric. The dimension weights are modified for each lab assignment to match the key objectives being emphasized in each assignment (see Table 3). Students are deemed to have successfully mastered a course objective when the class average on the final project is at least a 3 (Good) for the respective dimension on the grading rubric.

5 Conclusion

The proliferation of embedded devices and field-programmable gate arrays (FPGAs) that includes significant signal processing functionality suggest that students preparing to enter industry specializing in the fields of signal processing or computer architecture need to have a knowledge of hardware digital signal processing (DSP) implementations. A course for teaching the concepts of fixed-point DSP hardware design has been proposed. This course

Dimension	Not Yet	Beginning	Adequate	Excellent
Tools & Environment	No evidence of computing tool / environment use is present.	Minimal use of computing tools / environments is evident.	Appropriate use of computing tools / environments is evident.	Optimal use of appropriate computing tools / environments is evident.
Design	No or little design is apparent.	Basic elements of design are present.	Good design is demonstrated and documented.	Superior design is demonstrated and well documented.
Implementation	Implementation does not meet project requirements.	Project is poorly implemented and documented.	Implementation meets all project requirements and is documented.	Implementation is elegant and well documented.
Analysis	No or little analysis is apparent.	Minimal or ad-hoc analysis is present.	Effective analysis is present and documented.	Comprehensive analysis is present and well documented.
Creativity	No or little creativity is apparent.	Independent thought apparent but at a minimal level.	Creativity and independent thought shown.	Innovative and effective approach to problem solving demonstrated.
Cooperation with Team	Demonstrates a contentious attitude and/or does not fulfill commitments.	Normally demonstrates the ability to work with others and fulfills most commitments.	Demonstrates the ability to work with others and fulfills commitments.	Demonstrates a positive team spirit and fulfills all commitments.

Table 5: Laboratory Assignment Grading Rubric

emphasizes system-level design where students are introduced to the complexities of real-world problems and provided an opportunity to enhance their problem-solving and creative design skills.

References

- [1] L. S. DeBrunner and V. DeBrunner, "The case for teaching DSP algorithms in conjunction with implementations," in *Proc. IEEE Signal Processing Society's 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Sept. 2002.
- [2] A. J. S. Ferreira and F. J. O. Restivo, "Grasping the potential of digital signal processing through real-time DSP laboratory experiments," in *Proc. IEEE Signal Processing Society's 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Sept. 2002.
- [3] C. H. G. Wright, T. B. Welch, D. M. Etter, and M. G. Morrow, "Teaching DSP: bridging the gap from theory to real-time hardware," *Computers in Education Journal*, vol. 13, no. 3, pp. 14–26, July–September 2003.
- [4] T. S. Hall and D. V. Anderson, "From algorithms to gates: Developing a pedagogical framework for DSP hardware design," in *Proc. IEEE Signal Processing Society's 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Sept. 2002.
- [5] R. Chassaing and D. W. Horning, *Digital signal processing with C and the TMS320C25*. Wiley Interscience, 1990.
- [6] R. Chassaing, *Digital signal processing with C and the TMS320C30*. Wiley Interscience, 1992.
- [7] ———, *Digital signal processing: laboratory experiments using C and the TMS320C31 DSX*. Wiley Interscience, 2002.
- [8] W. T. Padgett, "An undergraduate fixed point DSP course," in *Proc. IEEE Signal Processing Society's 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Sept. 2002.
- [9] J. Vieira, A. Tome, and J. Rodrigues, "Providing an environment to teach DSP algorithms," in *Proc. IEEE Signal Processing Society's 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Sept. 2002.
- [10] A. J. Kornecki, "Real-time systems course in undergraduate cs/ce programs," *CD-ROM Supplement, IEEE Transactions on Education*, vol. 40, p. 9, Nov. 1997.
- [11] C. H. G. Wright, T. B. Welch, D. M. Etter, and M. G. Morrow, "Teaching hardware-based DSP: theory to practice," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, Orlando, FL, May 2002, pp. 4148–4151.
- [12] W. S. Gan, "Teaching and learning the hows and whys of real-time digital signal processing," *IEEE Transactions on Education*, vol. 45, no. 4, pp. 336–343, Nov. 2002.

- [13] S. L. Wood, "Signal processing and architecture in the lower division electrical engineering core," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, May 2001, pp. 2713–2716.
- [14] T. S. Hall and D. V. Anderson, "A framework for teaching real-time digital signal processing with field-programmable gate arrays," *IEEE Transactions on Education*, vol. 48, no. 3, Aug. 2005.
- [15] K. Newman, J. O. Hamblen, and T. S. Hall, "An introductory digital design course using a low-cost autonomous robot," *IEEE Transactions on Education*, vol. 45, no. 3, pp. 289–296, Aug. 2002.
- [16] J. O. Hamblen, "Rapid prototyping using field-programmable logic devices," *IEEE Micro*, vol. 20, no. 3, pp. 29–37, May/June 2000.
- [17] T. S. Hall and D. V. Anderson, "Merging theory and implementation: A framework for teaching DSP hardware design," in *Proceedings of the American Society for Engineering Education Annual Conference*, 2004.
- [18] L. D. Feisel and A. J. Rosa, "The role of the laboratory in undergraduate engineering education," *Journal of Engineering Education*, vol. 94, no. 1, pp. 121–130, Jan. 2005.
- [19] T. S. Hall and J. O. Hamblen, "System-on-a-programmable-chip development platforms in the classroom," *IEEE Transactions on Education*, vol. 47, no. 4, pp. 502–507, Nov. 2004.
- [20] *Simulink HDL Coder 1.0*, HTML File, Mathworks, <http://www.mathworks.com/products/slhdlcoder/>, Jan. 2007.
- [21] *LabVIEW FPGA*, HTML File, National Instruments, <http://www.labview.com/fpga/>, Jan. 2007.