FFT Spectrum Analyzer Project for Teaching Digital Signal Processing With FPGA Devices

Trini Sansaloni, Asun Pérez-Pascual, Vicente Torres, Vicenç Almenar, José F. Toledo, and Javier Valls

Abstract—**This paper presents a course on digital signal processing with field-programmable gate arrays (FPGA) devices. The course integrates two separate disciplines, digital signal processing (DSP) and very large scale integration (VLSI) design, and focuses on the development of a sophisticated DSP design from simulation to fixed-point implementation. The structure and methodology used in the proposed course are oriented to the design and implementation of an fast Fourier transform (FFT) spectrum analyzer. This application covers most topics included in a DSP course and gives better results that those obtained with typical courses performing independent multiple simple experiments. The project is divided into modules that show specific learning necessities and determine the course contents and organization. Each laboratory part is dedicated to design and implements the block of the analyzer related to the theoretical content presented in the class. At the end of the course the students have designed all the pieces in the DSP project and have completed and verified the system. The used methodology enables students and engineers to understand and develop complex fixed-point applications, looking for the best signal processing algorithms on hardware implementations, and also results in more motivated and active students.**

Index Terms—**Arithmetic, coordinate rotation digital computer (CORDIC), decimate filters, digital circuits, digital signal processing (DSP), direct digital synthesis (DDS), fast Fourier transform (FFT), field-programmable gate arrays (FPGA) implementation, logic design, mixer, real-time applications, signal processing, spectrum analyzer, windowing.**

I. Introduction

THE digital signal processing (DSP) industry requires engineers capable of developing DSP solutions to applications. Thus, DSP is an important component in a curriculum, especially when the real-time applications are physically implemented. Many curricula include separate classes in both DSP algorithms and hardware implementations. Finding a choice which combines both skills in a DSP hardware course is not easy. Usually, digital signal processing courses focused on the implementation issues use DSP microprocessors to implement the algorithms [1]–[8]. Hardware DSP courses are seldom employed because of lack of time and resources [9], [10]. However, with the development of field-programmable gate arrays (FPGA) devices and software development tools, teaching most efficient hardware DSP algorithms from design to implementation can be accomplished effectively [11]. By using FPGAs, students are given the opportunity to design and explore custom hardware implementations.

A framework for teaching real-time digital signal processing with FPGAs is presented in [11]. However, two different environments are used: Matlab software for modeling the system and a hardware description language (HDL) for performing hardware implementation on FPGA. The two environments result in a nonfast prototyping system for teaching real-time DSP implementations. With the above issues in mind, a course on FPGA digital signal processing was introduced into the postgraduate curriculum. This course introduces the use of system generator (SG) [12] environment to model DSP projects. Its capability to model complex DSP systems and to transform the theoretical design into a finite precision fixed-point system are two important reasons in this choice [13]. This fast system prototyping tool allows students to see the effects of their design decisions.

This course, which emphasizes practical DSP aspects, is based on the development of a real-world project on FPGA devices. The design of a complex project is used to motivate students, to affirm course contents, and to provide greater progress in technical skills and abilities that the traditional laboratory practices used earlier in the same course.

The fast Fourier transform (FFT) spectrum analyzer project has been chosen to relate all the topics in the syllabus and to put them into context: hardware architectures for DSP algorithms, arithmetic circuits, FPGA devices and their resources to implement DSP blocks, and software tools for fast system prototyping. Each of the blocks of this system is a good excuse to explain one of the modules in a DSP course and an excellent practice to develop in the laboratory. Other applications have been considered to be implemented; however, for DSP teaching purpose, the spectrum analyzer project is the best candidate.

The rest of the paper is organized as follows. Section II presents the course details, their logistic, contents, and organization. Section III describes the pedagogical issues of the course (goals and methodology). Section IV is focused on the necessary details to implement the FFT spectrum analyzer, in particular, its blocks and the required development tools. Finally, the evaluation data is addressed, and the conclusions are presented.

II. Course Details

A. Course Logistics

This course is a postgraduate course taught by the Digital Communications Laboratory group within the Electronics Engineering Department at Polytechnic University of Valencia,

Manuscript received April 5, 2006; revised March 21, 2007.

T. Sansaloni, A. Pérez-Pascual, V. Torres, and J. Valls are with the Department of Electronics, Universidad Politécnica de Valencia, 46730 Grao de Gandia, Spain (e-mail: tmsansal@eln.upv.es).

V. Almenar is with the Department of Communications, Universidad Politécnica de Valencia, 46730 Grao de Gandia, Spain.

J. F. Toledo is with the Electronics Engineering Department, Universidad Politécnica de Valencia, 46730 Grao de Gandia, Spain.

Digital Object Identifier 10.1109/TE.2007.900025

TABLE I
TIME SCHEDULE FOR CONDUCTING A DSP DESIGN COURSE

Module	Hours used	Topic	Lab
	1h	Introduction to DSP on FPGAs (1h)	--
1	4h	Spectrum analyzer: system properties and function block diagram (2h)	Ideal system model (2h)
2	3h	Number representations and fixed-point arithmetic. FPGA resources for DSPs (2h)	Windowing block (1h)
3	4h	Finite word-length effects. Arithmetic circuits for FPGA (2.5h)	Constant coefficient multipliers design (1.5h)
4A	4h	Digital filters: FIR, multi-rate (3h)	Half-band decimated filters (1h)
4B	4h	Digital filters: IIR (2h)	Half-band decimated filters (2h)
5	4h	Fast Fourier Transform (2h)	FFT (2h)
6	4h	Direct Digital Synthesis (1.5h)	Mixer design and implementation (2.5h)
7	4h	CORDIC algorithm (2h)	Cartesian-to-polar converter and log operator (2h)
	10h	--	Integration and verification of the project (10h)

Spain, and it is called Digital Signal Processing on FPGAs. The course has been successfully taught with the proposed methodology during the last two years. Earlier, it was a typical course with independent experiments.

This course is designed for students specializing in electronics and specializing in DSP. However, in practice, the majority of students enrolling in this course have been electronics students. Students taking this course are expected to be familiar with Matlab, Simulink, and ISE Xilinx tools. Previous knowledge of DSP theory and FPGA implementation of digital circuits is also required.

This course is scheduled for 12 weeks with an average of 4 h of class each week. The distribution between hours of lecture and hours of laboratory depends on the module. The course has a length of 42 h: 18 h are dedicated to theory, and 24 h are spent in the laboratory to develop the FFT spectrum analyzer project.

Additional work at home is needed to mature the theoretical concepts and to complete the laboratory work. This extra work is estimated not to exceed 20 h. The time schedule is listed in Table I.

There are no examinations in this course. Grading is based on the demonstration and evaluation of the design and on the quality of a written report. The project must be completed at the end of the quarter.

B. Course Contents

As mentioned earlier, to improve the understanding of the DSP course contents and increase the motivation of the students, the course is oriented to solve a realistic DSP application: "the FFT spectrum analyzer." The design specifications, the block diagram of the spectrum analyzer, and an overview of the DSP topics are needed to understand the structure of the course.

Each module in the spectrum analyzer block diagram shows a specific learning necessity, which determines the organization of the course contents in seven modules. Each block corresponds to a module as is shown in Fig. 1. Each class has both a theoretical and a laboratory part. The purpose of the laboratory part is to complement the theory with the design and implementation of one of the different blocks of the spectrum analyzer. The division of the project into smaller parts gives sense to the contents, helps students to solve a complex project, and increases the motivation of the students.

Table II summarizes the topics covered in each module and the work proposed for the laboratory. The contents of each module are briefly detailed below. Although written documentation has been generated for the course, some books are referenced as a bibliography to study in depth [14]–[19].

C. Course Development

The first day is devoted to introducing the syllabus, the goals, the methodology that is going to be applied, and how students will be evaluated.

In the first module the project is introduced, and the spectrum analyzer block diagram is explained, highlighting the connections between each block and the course contents. Topics, such as frequency translation, multirate filtering, windowing, FFT, and the coordinate rotation digital computer (CORDIC) algorithm, are briefly described. Most of these concepts are not new to the students, so that the only requirement is to give

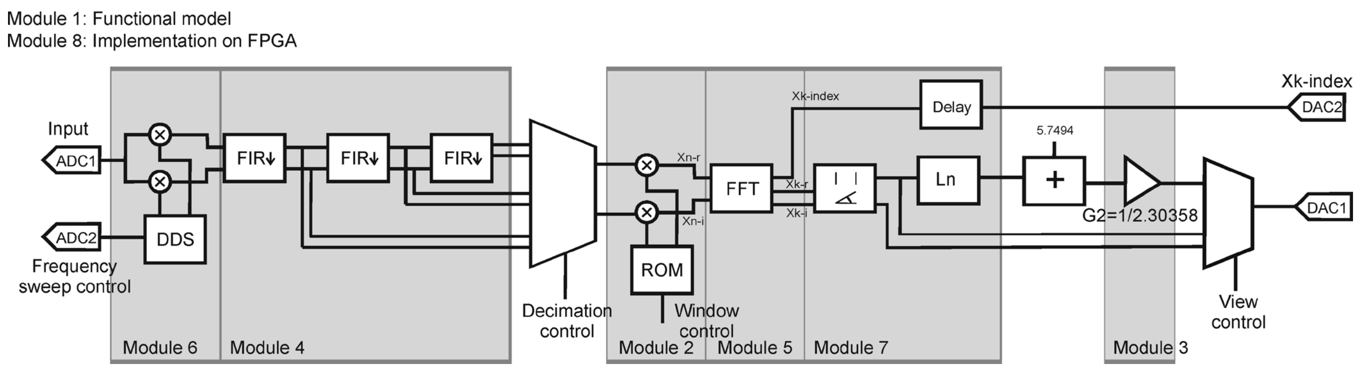

Fig. 1. Block diagram of the spectrum analyzer.

TABLE II
SPECTRUM ANALYZER SPECIFICATIONS

FREQUENCY
Frequency range: 0-30 MHz
Frequency resolution: 512 lines
FFT Spans: Continuous shift frequency
Zoom: x2, x4
AMPLITUDE
Signal amplitude: ±1V
Dynamic range: 75 dB
ADC resolution: 14 bits
GENERAL
ADC sampling rate: 65 MHz
System clock frequency: 65 MHz
Windows: Uniform, Blackman-Harris, Hanning, Flat-top
Views: Linear magnitude, logarithmic magnitude, phase

a system overview and to polish up these concepts. The laboratory for this class focuses on modeling the spectrum analyzer with Simulink, ignoring quantization and overflow effects and avoiding fixed-point arithmetic. The result is a nonimplementable model that will allow the students to understand the project and test the new subsystems that they will design in the coming classes in the context of the whole system.

The second module works with hardware resources available in the FPGA devices that can be used to implement DSP algorithms and with how these resources can be inferred from the SG. Specifically, the following FPGA resources are taken into account: lookup tables (LUTs) to implement combinatorial logic, distributed RAM or ROM memory and shift registers, multiplexers to enlarge the LUT-based combinatorial logic or memory, embedded dual-port RAM memories which carry propagation chains to implement arithmetic circuits, and embedded two's complement multipliers. The functions and limitations of these resources are discussed. Emphasis is placed on how to achieve the different functions from the SG tool and how the SG infers hardware if a function exceeds the limitation of a resource. In the laboratory for this class the students have to model with the SG, using FPGA resources, such as dual-pot RAM and multipliers, the windowing block for the spectrum analyzer.

The third module covers the fundamentals of the fixed-point arithmetic and the basic arithmetic circuits for FPGA. The quantization and overflow methods and the effects of fixed-point arithmetic and their effect in the hardware implementation are explained. After the study of these effects different ways of implementing the basic arithmetic circuits, adders, and multipliers are detailed. Ripple-carry and carry-save techniques are presented and evaluated for FPGA. Two methods for constant coefficient multiplication are explained: 1) using the canonic signed-digit codification of the coefficients to implement wired multipliers and 2) using tables and tree-adders to store and add the partial products. The laboratory for this lesson comprises the design of constant coefficient multipliers in the context of the spectrum analyzer implementation. Specifically, the design of multipliers to compensate the gain of CORDIC blocks is described below.

The fourth module is focused on the implementation of digital filters. Therefore, filter design methods, structures, and quantization effects are reviewed for both finite-impulse-response (FIR) and infinite-impulse-response (IIR) filters. Two strategies are presented for implementing constant coefficient filters: wired filters based on sharing subexpression and filters based on distributed arithmetic. Look-ahead techniques of IIR filter pipelining are briefly introduced. Multirate filters are also considered in this module, so that after polishing up decimation and interpolation concepts, polyphase implementation architectures are explained. Special cases, such as half-band filters are also discussed. As previously noted, the spectrum analyzer needs decimation filters after the mixer. These filters are designed in the laboratory using the filter design and analysis tool (FDATool) software [20].

The spectrum analyzer uses a FFT to obtain the spectrum components of the signal, thus the fifth module covers the implementation issues of the FFT algorithm. FFT is shown to be based on the repetition of a simple structure called butterfly. The two approaches to implement butterfly processors (decimation in time and decimation in frequency algorithms) are introduced, and the radix-2, radix-4, radix-8, and split-radix butterflies are compared in terms of hardware resources. The complex-number multiplication is also discussed as the most important computational element of the butterflies. The different FFT architectures (array, column, pipeline, and single processor approaches) are presented. A comparative table, in terms of hardware cost and computation speed, for these approaches is shown. Finally, the Xilinx FFT Core [21] parameters are enumerated, and the SG's FFT core use is described. The laboratory for this module comprises: 1) the FFT core configuration according to the real-time spectrum analyzer's specifications and 2) the generation of the enable signal for the FFT block (required to adapt the processing frequency to the actual decimation rate).

The sixth module works with direct digital synthesis (DDS), which is introduced using a direct LUT-based architecture, as a method of generating waveforms in the digital domain. The sources of spurious generation and the amplitude and phase truncation are described. A couple of memory compression and interpolation techniques are presented, and, finally, the DDS based on the CORDIC algorithm is explained. For this purpose, CORDIC is only considered as a phase-to-amplitude converter, and the implementation details are not given yet. The laboratory work for this module proposes the development of the DDS required for the spectrum analyzer to perform the spectrum frequency shift. The design must also be implemented and verified on the target DSP hardware.

The last module considers the CORDIC algorithm, which is introduced as an iterative algorithm that only requires adder-substractors and shifters. Rotation and vectoring operation modes and the convergence range and precision of CORDIC are explained, and the fully parallel architecture is shown. Finally, the algorithm is generalized to operate in circular, hyperbolic, and linear coordinates systems, and the operation $\log(x)$ is studied. In the laboratory for this module a CORDIC is used for rectangular to polar conversion of the transformed values and for the computation of the natural logarithm to represent the spectrum in a logarithmic scale.

When the students complete the course, they have modelled most of the blocks they need to implement the entire FFT spectrum analyzer and know the design flow and tools. Some additional laboratory sessions are needed to put all the pieces together and to verify their behavior on hardware. Fig. 2 shows the implemented spectrum analyzer.

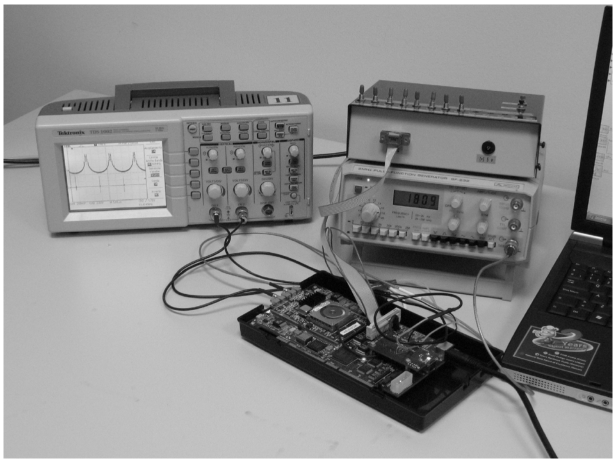
Fig. 2. Spectrum analyzer on FPGA.

III. Goals and Methodology

The main objective of the course is to familiarize students with the fundamental principles of the design and implementation of DSP algorithms on FPGA devices. Several fields can be distinguished within this area: DSP theory, hardware architectures to implement DSP algorithms, binary arithmetic, specific hardware resources in FPGA devices to implement DSP blocks, and software tools for fast system prototyping.

Decisions on algorithms, architecture, and arithmetic have a great influence on the performance achieved by the implemented hardware. These decisions have to be taken, considering that the hardware resources in the FPGA are fixed. Thus, to achieve the best performance, a software tool that allows the students to study these tradeoffs is needed.

The goal of the course is to accomplish the previous objective by developing a real-world DSP project from the block diagram to a real hardware implementation. The design of a fully digital spectrum analyzer has been selected as the target project. This project is an application that allows covering the basic topics in a course on DSP on FPGAs. The theory is focused on DSP concepts as sampling, frequency shifting, filter-design, decimation, windowing, FFT, and direct digital synthesis. It places emphasis on the hardware implementation architectures, on the design of arithmetic operators, and special schemes, such as distributed arithmetic or CORDIC algorithm on the hardware resources for DSP available in the target FPGA, a Xilinx Virtex-II device. The laboratory reinforces the theory and develops the skills to use simulation and FPGA implementation tools. Besides, designing and implementing a complex design is much more challenging for the students.

The course is designed to improve the understanding of the contents and to increase the motivation of the students, thus favoring learning. Although understanding and motivation are factors that influence mutually, the characteristics that are taken as objectives for the course relating to both factors are as follows.

- Understanding
  - — Course contents are presented as an answer to a learning necessity—a realistic spectrum analyzer project introduced at the beginning of the course. The search for answers gives sense to the contents.
  - — The theoretical content is easily referenced in the frame provided by the base project.
  - — Written documentation has been generated (slides and textual information) for most theoretical parts of the course.
  - — The dynamics of the classes allow each student to advance at his/her own learning pace, thus easing the assimilation of the concepts.
  - — A long practice time tries to foment the active attitude of the students. In the theoretical parts of the sessions instructors take special care over the type of questions asked to the students and the management of the answers.
- Motivation
  - — The proposed base project is not purely academic, but realistic and interesting.

— The activities in the sessions are predominantly practical, but alternate lectures and practical work to vary the activity and thus renew the students' attention.
— The work is developed in pairs rather than individually, although the available infrastructure allows both options.
— The students enjoy autonomy and decision-making in the practical parts.
— At the end of course the students are conscious of having fulfilled the objectives, since they have satisfactorily developed the base project.

IV. Target Project Details

A. The Target Project

The FFT spectrum analyzer is the real-world application chosen to consolidate the theoretical concepts. The target application covers most of the topics presented in the course. The main functional blocks in the FFT spectrum analyzer are represented in Fig. 1. The spectrum analyzer is composed of a digital mixer based on a direct digital synthesizer and multipliers, a cascade of decimators (FIR filters), a multiplexer, a block to perform the windowing based on a ROM and two multipliers, an FFT core, and two blocks to compute the modulus and logarithmic function based on the CORDIC algorithm.

The spectrum analyzer specifications are summarized in Table I. The student has to make decisions on the order of the filters, the data-path precisions for the different modules in the design, and the architectural options for implementing each DSP block. The spectrum analyzer must be optimized in the area for the given specifications.

The input signal is an analogue voltage between -1 and 1 V, which is filtered through an anti-aliasing filter (on the XtremeDSP board) and converted to digital with a 14-b analog-to-digital converter (ADC). The signal processing after this point is performed in the FPGA device.

The first DSP stage is a digital mixer required to perform a frequency shift in the input signal spectrum according to the frequency generated by the direct digital synthesizer. In order to choose the spectrum resolution, a cascade of three half-band decimation filters is used. Therefore, the spectrum span (frequency range) can be controlled at the first stages in the design. The FFT module is the heart of the analyzer. The length of the transform determines the frequency resolution. This block computes the FFT algorithm and returns the complex transformed values. To display the magnitude of the spectrum, additional modules must be included. A CORDIC processor is used to compute the modulus of the complex signal. Measurements of magnitude are typically shown in logarithmic scale using $\mathrm{dB_V}$ units. This conversion is made by another CORDIC processor. Finally, the processed signal is converted into an analogue signal and displayed on an oscilloscope.

The gain of the CORDIC core used to compute the modulus and the scaling in the FFT are compensated by adding a constant to the output (Fig. 1). Since SG libraries do not include a log 10 module, this operation is performed by using a natural logarithm and adding an additional multiplier by a constant. The output of the system can be rewritten in the way shown in equation 1

$$\begin{aligned}
&\log_{10}(\mathrm{out} * c_{\mathrm{FFT}} * c_{\mathrm{CORDIC}})\\
&\quad = \log_{10}(\mathrm{out} * 512 * 0.6072)\\
&\quad = \frac{\ln(\mathrm{out}) + \ln(512 * 0.6072)}{\ln 10}\\
&\quad = (\ln(\mathrm{out}) + 5.7394) \cdot \frac{1}{2.30358}.
\end{aligned}$$

B. DSP Modeling and Implementation Environment

The SG [12] environment is employed to model the DSP project. Some reasons are important in this choice. Its capability to model complex DSP systems and to transform the theoretical design into a finite precision fixed-point system are two of them [13]. The SG eases FPGA hardware design by providing access to architectural descriptions of the algorithms and allowing the manipulation of data-path precision to obtain efficient FPGA implementations. The libraries provided with the tool offer a wide range of elements that simplify the design process. The tool automatically generates an FPGA implementation project for the model. The elaborated project can then be placed and routed in an FPGA by using Xilinx ISE tools.

The system must be modelled, simulated with floating-point data, and re-simulated with quantified fixed-point data. Then the design has to be verified and tested on a programmable device. To perform these tasks software and hardware tools have to be employed.

Some of the design specifications, such as dynamic range and signal amplitude, are given by the chosen hardware platform—the Xtreme DSP kit of Nallatech. This hardware consists of a BenONE motherboard [22] and a BenADDA module [23] that is plugged onto a slot on the BenONE. The platform provides two 14-b 65-mega-sample-per-second (MSPS) ADCs, two 14-b 160-MSPS digital-to-analog converters (DACs), and one Virtex-II FPGA device (XC2V6000-FF1152). An external clock or an on-board 65 MHz oscillator can be used as system clock and ADC–DAC clock.

In the project (Fig. 1), the ADC1 is used to capture the signal to be analyzed and the ADC2 captures a word that controls the value of the frequency shift for the mixer. The magnitude of the obtained spectrum is connected to DAC1, while the index of the transformed samples is connected to DAC2. The spectrum is displayed on an oscilloscope by connecting the DAC2 and DAC1 outputs to the oscilloscope in X–Y mode.

The design flow uses Simulink and SG software to create a high-level model of the hardware system. The SG is a system-level modeling tool that allows the creation and simulation of real hardware in the Simulink environment. The software provides additional Xilinx components libraries to Simulink, such as math functions, DSP functions, and the model of other specific resources in the board, such as ADC and DAC converters. The SG also performs the generation of the Xilinx ISE project files (VHDL files and constraints file) required to generate the FPGA programming file.

TABLE III
EVALUATION OF THE STUDENTS WHO PARTICIPATED IN THE COURSE

Question posed / Description	Average* 2005	Average* 2004
The course organization and planning are adequate.	8.9	6.7
The level of the course is adequate.	8.9	7.5
The contents of this course are valuable.	8.3	7.5
The contents of this course are well-developed.	8.3	8.3
The generated documentation for the course is useful and adequate.	7.9	8.3
The active participation of students in the class is strongly encouraged.	9.2	7.5
The contents of the course are focused on many examples and professional applications.	9.2	7.1

The models developed with the SG allow the designer the direct implementation of fixed-point arithmetic versions of modelled algorithms. The implementation on FPGA has been done with Xilinx ISE software, working with the project generated by the SG. Finally, the field upgradeable systems environment (FUSE) is used to configure and control the FPGA.

V. EVALUATION DATA

From the teaching assessment over the past few semesters, 100% of the students are able to understand and use the concepts of fixed-point signal processing design and implement the different blocks of the spectrum analyzer efficiently. Almost 90% of the students have successfully completed their spectrum analyzer and even ported their design to the hardware. Overall, learning objectives proposed in the course have been met, and the method under evaluation has been well received by the students.

The experience reveals that the used methodology results in a better understanding of the theory and in more motivated and active students. To evaluate the course, students are invited to complete an end-of-course survey that provides an important source of feedback. Table III shows the main results of the last two years. Survey results revealed an agreement or strong agreement with the statement "the contents of this course are valuable." Working with a ten-point scale, where "10" signifies strong agreement, the average value of 8.3 was given for this item. The item "the contents of this course are well-developed" had a value of 8.3. Better results were obtained in items related to the teaching method; items such as "the active participation of students in the class is strongly encouraged" and "contents of the course are focused on a lot of examples and professional applications" had a value of 9.2. Other items also provided very positive evaluations.

VI. FINAL REMARKS AND CONCLUSIONS

In this paper a DSP on a FPGA course has been presented. The objective of the course is to show the principles of the design and implementation of DSP algorithms on FPGA devices. The course is based on the design of a DSP project—a fully digital spectrum analyzer. This paper summarizes the contents of the course placing emphasis on its methodology and organization. Most of the algorithms presented in the classroom are used in the project and implemented on FPGAs using the SG software tool.

The proposed educational approach allows students to develop their abilities to design and solve applications meeting desired specifications. This objective is not met when small independent laboratory exercises are performed [5]–[8]. Students are encouraged to experiment, analyze, and interpret data; these abilities are stronger developed when the project complexity increases. This course also offers the possibility of fast prototyping by using modern developing tools; other courses use DSP processors instead of FPGAs [1]–[8]. The course reported in [11] needs to develop specific interface and requires more complex design description (VHDL), therefore, more developing time. The experience reveals that the proposed methodology results in a better understanding of the theory and in more motivated and active students.

ACKNOWLEDGMENT

The authors would like to thank the Xilinx University Program for donating the software tools used in the course.

REFERENCES

[1] W. T. Padgett, "An undergraduate fixed point DSP course," in *Proc. IEEE Signal Processing Society 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Oct.. 2002, pp. 302–305.
[2] J. Vieira, A. Tome, and J. Rodrigues, "Providing an environment to teach DSP algorithms," in *Proc. IEEE Signal Processing Society 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Oct. 2002, pp. 292–296.
[3] A. J. Kornecki, "Real-time systems course in undergraduate CS/CE programs," *IEEE Trans. Educ.*, vol. 40, no. 4, p. 9, Nov. 1997, CD-ROM supplement.
[4] C. H. G. Wright, T. B. Welch, D. M. Etter, and M. G. Morrow, "Teaching hardware-based DSP: Theory to practice," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, Orlando, FL, May 2002, vol. 4, pp. 4148–4151.
[5] W.-S. Gan and M. Kuo, "Teaching DSP software development: Form design to fixed-point implementations," *IEEE Trans. Educ.*, vol. 49, no. 1, pp. 122–131, Feb. 2006.
[6] W.-S. Gan, "Teaching and learning the hows and whys of real-time digital signal processing," *IEEE Trans. Educ.*, vol. 45, no. 4, pp. 336–343, Nov. 2002.
[7] W.-S. Gan, Y.-K. Chong, W. Gong, and W.-T. Tan, "Rapid prototyping system for teachin real-time digital signal processing," *IEEE Trans. Educ.*, vol. 43, no. 1, pp. 19–24, Feb. 2000.
[8] 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 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Oct. 2002, pp. 286–291.

[9] L. S. DeBrunner and V. DeBrunner, “The case for teaching DSP algorithms in conjunction with implementations,” in *Proc. IEEE Signal Processing Society 2nd Signal Processing Education Workshop*, Pine Mountain, GA, Oct. 2002, pp. 195–200.

[10] 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,” *Comput. Educ. J.*, vol. 13, no. 3, pp. 14–26, Jul.–Sep. 2003.

[11] T. S. Hall and D. V. Anderson, “A framework for teaching real-time digital signal processing with field-programmable gate arrays,” *IEEE Trans. Educ.*, vol. 48, no. 3, pp. 551–558, Aug. 2005.

[12] *Xilinx System Generator for DSP, v 8.1 User’s Guide,* Xilinx, Inc., San Jose, CA.

[13] D. Turney *et al.*, “Modeling and Implementation of DSP FPGA Solutions,” White Paper, 2000.

[14] C. F. Coombs*, Electronic Instrument Handbook*. New York: McGraw-Hill, 2000.

[15] A. V. Oppenheim, R. W. Schafer, and J. R. Buck*, Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[16] U. Meyer-Baese*, Digital Signal Processing With Field Programmable Gate Arrays*. Berlin, Germany: Springer-Verlag, 2001.

[17] K. K. Parhi*, VLSI Signal Procesing System*. New York: Wiley, 1999.

[18] L. Wanhamar*, DSP Integrated Circuits*. New York: Academic, 1999.

[19] B. Parhami*, Computer Arithmetic. Algorithms and Hardware Design*. London, U.K.: Oxford Univ. Press, 2000.

[20] *Signal Processing Toolbox User’s Guide,* MathWorks, Inc., Natick MA, 2006.

[21] Fast Fourier Transform v3.2 Xilinx Core. DS260 Aug. 2005, Xilinx Product Specifications [Online]. Available: www.xilinx.com

[22] Nallatech Ltd., BenOne product brief, 2002.

[23] Nallatech Ltd., BenAdda product brief, 2002.

Trini Sansaloni received the M.Eng. and Ph.D. degrees in telecommunication engineering from the Universidad Politecnica de Valencia, Grao de Gandia, Spain, in 1994 and 2001, respectively.

She is an Associate Professor in the Department of Electronics at the Universidad Politecnica de Valencia. Her current research interests include the design of FPGA-based systems and VLSI signal processing.

Asun Pérez-Pascual received the M.Eng and Ph.D. degrees in telecommunication engineering from the Universidad Politecnica de Valencia, Grao de Gandia, Spain, in 1997 and 2002, respectively.

She has been an Associate Professor in the Department of Electronics at the Universidad Politecnica de Valencia since 2002. Her current research interests include the design of FPGA-based systems, computer arithmetic, VLSI signal processing, and digital communications.

Vicente Torres received the M.Eng. and Ph.D. degrees in telecommunication engineering from the Universidad Politecnica de Valencia, Grao de Gandia, Spain, in 1994 and 2001, respectively.

He has been an Associate Professor in the Department of Electronics at the Universidad Politecnica de Valencia since 1995. His current research interests include the design of FPGA-based systems with a focus on digital communications.

Vicenç Almenar received the M.Eng. and Ph.D. degrees in telecommunication engineering from the Universidad Politecnica de Valencia, Grao de Gandia, Spain, in 1993 and 1999, respectively.

He is currently an Associate Professor in the Department of Communications at the Universidad Politecnica de Valencia. In 2000, he spent five months at the Center for Communications Systems Research, University of Surrey, Guildford, U.K., where he was involved in research on digital signal processing for digital communications. His current research interests include OFDM, MIMO, signal processing, and simulation of digital communications systems.

José F. Toledo received the M.Eng. and Ph.D. degrees in telecommunication engineering from the Universidad Politécnica de Valencia, Grao de Gandia, Spain, in 1995 and 2002, respectively.

He researched his Ph.D. dissertation at the European Laboratory for Particle Physics (CERN), Geneva, Switzerland, from 1998 to 2001, defining and developing data acquisition systems for high-energy physics experiments. He is currently a Lecturer in the Electronics Engineering Department at the Universidad Politécnica de Valencia. His research interests are in the area of readout and data acquisition electronics for nuclear medicine and particle physics applications.

Javier Valls received the M.Eng. degree in telecommunication engineering from the Universidad Politecnica de Cataluña, Barcelona, Spain, in 1993 and the Ph.D. degree in telecommunication engineering from the Universidad Politecnica de Valencia, Grao de Gandia, Spain, in 1999.

He has been an Associate Professor in the Department of Electronics at the Universidad Politecnica de Valencia since 1996. His current research interests include the design of FPGA-based systems, computer arithmetic, VLSI signal processing, and digital communications.