PROCEEDINGS OF SPIE

Independent Component Analyses, Wavelets, Unsupervised Nano-Biomimetic Sensors, and Neural Networks VI

Harold H. Szu
F. Jack Agee
Editors

17–19 March 2008
Orlando, Florida, USA

Volume 6979

SPIE

Contents

DSP with FPGAs: A Xilinx/Simulink-based Course and Laboratory

Uwe Meyer-Baese[a], A. Vera[b], A. Meyer-Baese[a], M. Pattichis[b], R. Perry[a],
[a]FAMU-FSU, ECE Dept., 2525 Pottsdamer Street, Tallahassee, FL USA-32310;
[b]University of New Mexico, ECE Dept., Albuquerque, NM 87131

ABSTRACT

This is the third paper in a series that introduces a MatLab/Simulink-based design flow for FPGAs at an undergraduate curriculum level. In the first paper presented at SPIE 2006 we analyzed the design tools, while in the second paper presented at SPIE 2007 we reported on the appropriate topics for the lectures and labs. In this third paper we first give an overview based on the 12-year EDA cycle on why FPGAs have now reached a level where SOPC design is possible and why MatLab/Simulink is favored by both leaders in the FPGA field: Altera (DSP builder) and Xilinx (System Generator). We then describe the Xilinx Blackboard educational material development (EMD) that has been used in Spring 2007 and Spring 2008 to teach a Xilinx System Generator based course and laboratory.

Keywords: Field Programmable Gate Array (FPGA), Simulink, undergraduate curriculum, Xilinx System Generator, Digital Signal Processing (DSP), educational material development (EMD), CCLI.

1. INTRODUCTION

Every 18 months the number of transistors on a DRAM memory chip doubles; equivalently, every 3 years the number of transistors increases by a factor of four. When Gordon Moore[1], co-founder and former CEO from Intel, proposed this law back in 1965, he did not have many data to justify this empirical law. However, after 5 decades we now have seen that the law has accurately predicted transistor growth through time. There are many effects this law has on the development of equipment (e.g. microprocessors, low power portable devices, digital TVs, radios, etc.), and we would like to see how these have influenced the use of design tools for programmable logic devices over the years and, as a consequence, describe the currently needed EMD at the undergraduate level. The described EMD will concentrate on the Blackboard[2] (BB) web-based course material, while previous papers evaluated the design tools[3] and the topics to be included[4].

2. THE NEW DESIGN WAVE: MatLab/Simulink FPGA Design

Roughly every 12 years (i.e., a factor of 4^4=256 in transistor count) there is a major design paradigm change[5]. These different design waves are shown in Fig. 1. We can roughly say that for programmable logic devices we have seen about 4 design waves, alternating between graphical and text based designs.

1. In the first phase, Programmable Array Logic (PAL) or Generic Array Logic (GAL) which only contained a few AND/OR product terms were programmed with text-based PLD tools such as PALASM, ABEL, or CUPL[5]. These languages consisted of a simple equation that allowed editors to program the PAL-like devices. A part of such a GAL16V8 is shown in Fig. 2.

2. The simple equation editors used in the PAL assemblers had limits when designing larger blocks like a 16-bit adder. The number of AND/OR product terms then became too large to be described with the PAL assembler. At the same time FPGA devices were introduced which consisted of medium-size granularity logic blocks (i.e., 4-5 inputs, 1-2 output plus a 1-bit register. See Fig. 3) that could more easily be described with a graphical design tool, such as MaxPlusII from Altera or the Xilinx Foundation Tools, than a PAL assembler. At the end of this second PLD design tool wave, devices had reached about 65K equivalent logic gates.

3. One major problem in the graphical system design was that if the design got larger usually some kind of controller (FSM) unit was required. For this reason, most graphical tools included a "black-box" that allowed to

Independent Component Analyses, Wavelets, Unsupervised Nano-Biomimetic Sensors, and Neural Networks VI, edited by Harold H. Szu, F. Jack Agee, Proc. of SPIE Vol. 6979, 697907, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.776942

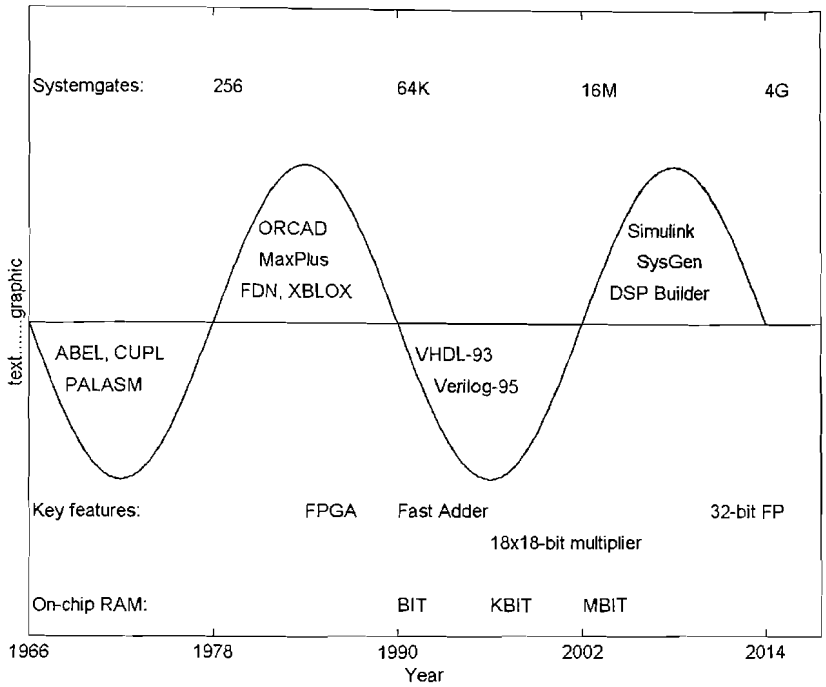


Figure 1: The FPGA design wave. The DRAM transistor count is typically about a factor 15 higher than available FPGA system gates.

import a PAL assembler's description of logic. The next state of the equation for the FSM, for instance, could be written with a simple equation editor. This somewhat awkward design style then led to the introduction of hardware description languages (HDL) that allowed the usage of one language only for data and the control path. The HDL was so important to the hardware designer that two IEEE standards were developed: VHDL 1087 and Verilog 1364. Another interesting feature of the HDL-based approach was the possibility of design reuse. Text-based design enabled the reuse of similar designs, i.e., one could write generic code that allowed, if a 16-bit adder was designed today and a 32-bit adder was needed tomorrow for instance, to change a single generic variable, while in a graphic design one would need a complete redesign. It was also impossible to switch device vendors, since the graphic libraries were not compatible between Altera's MaxPlus and Xilinx Foundation tools, for example. The design in HDL could more easily be moved across these vendor boundaries. Altera and Xilinx introduced the Quartus II and ISE software, respectively, to handle both HDL standards, i.e., to support VHDL and Verilog since, based on the geographic location, designers prefer different HDL languages. At the end of the third design wave (around 2002) FPGAs with 16M gates were available. These devices now had many logic blocks, though still about the same basic logic cell architecture as the first generation FPGAs from 1985. Additional features such as fast adder logic and embedded 18x18-bit array multipliers were included. On-chip memory has also increased accordingly as the data path width of the designs increased over the years (See Fig. 1.)

4. Currently we are at the beginning of a new design wave as Altera and Xilinx have introduced a MatLab/Simulink design flow. There are several reasons why this is a promising approach, but we think the most important reasons are as follows:

   - Without detailed knowledge of HDL or FPGA devices, the system can be simulated bit precise in MatLab and then automatically translated with a compiler into VHDL and a programming bit stream for the device.
   - The quality of the design of the high level tools is excellent and can exceed hand-based VHDL coding.
   - The tedious task of specifying testbench signals can now be done much easier using Simulink.

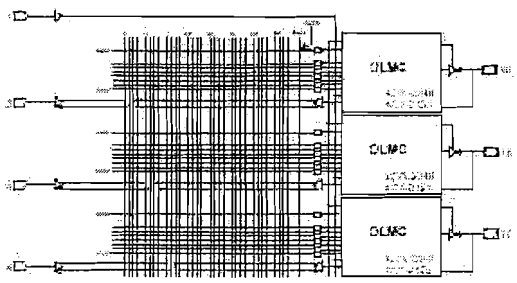


Figure 2: The GAL16v8 AND/OR matrix with output logic macro cell (OLMC).

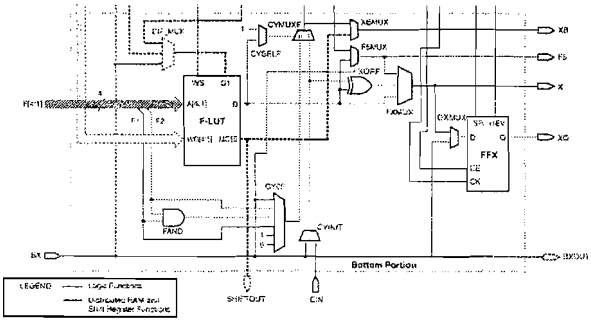


Figure 3: Core block elements of FPGAs: 4 input LUT, fast carry logic and flip-flop.

At the end of this design wave, in 2014, we expect to have powerful FPGAs with 4G system gates that allow us to build true systems-on-a-programmable-chip (SOPC) which support, for instance, a full featured 32-bit IEEE floating-point standard.

We think that at the undergraduate level, students should already be exposed to this important new design flow. We would like to report our progress on the development of an undergraduate course through the following notes, quizzes, and laboratory exercises, prepared for a Xilinx-based MatLab/Simulink design flow.

3. LECTURE MATERIAL DEVELOPMENT

In the first year of this study, we identified the appropriate topics from the textbook on DSP with FPGAs[6] for an undergraduate course and selected appropriate lab exercises. We used 11 consultants to evaluate appropriate topics and used the results to develop the lecture notes, quizzes, and labs. Table 2, below, and Fig. 3, from our previous paper[4], provide an overview of the topics in the Xilinx course for Spring 2007. In Fig. 3, from our previous paper[4], the thick grey line shows the actual sequence we used in both Altera and Xilinx courses, while the solid black line shows possible flows. Notice the three sections consisting of introductory material (tutorials), the core elements, and optional material (advanced topics) that can be selected based on the instructor's preference and available time. Following is a look at the developed EMD available online[2] for the students.

After the students are logged onto the course webpage, they are greeted by a general announcement, our customized banner, and buttons that navigate through the other resources. Later this first page will include announcements, for instance, for new posted quizzes or exam schedules. This welcome page is shown in Fig 4. The course map of all BB resources is shown in Fig. 5.

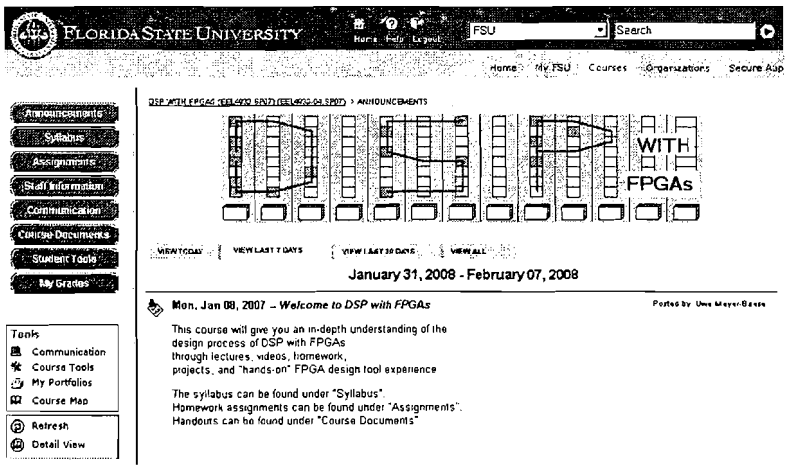


Figure 4: The welcome page.

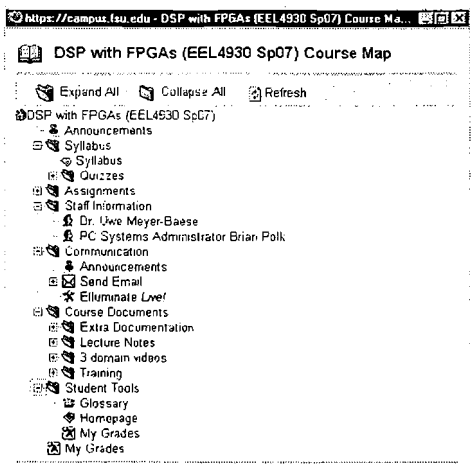
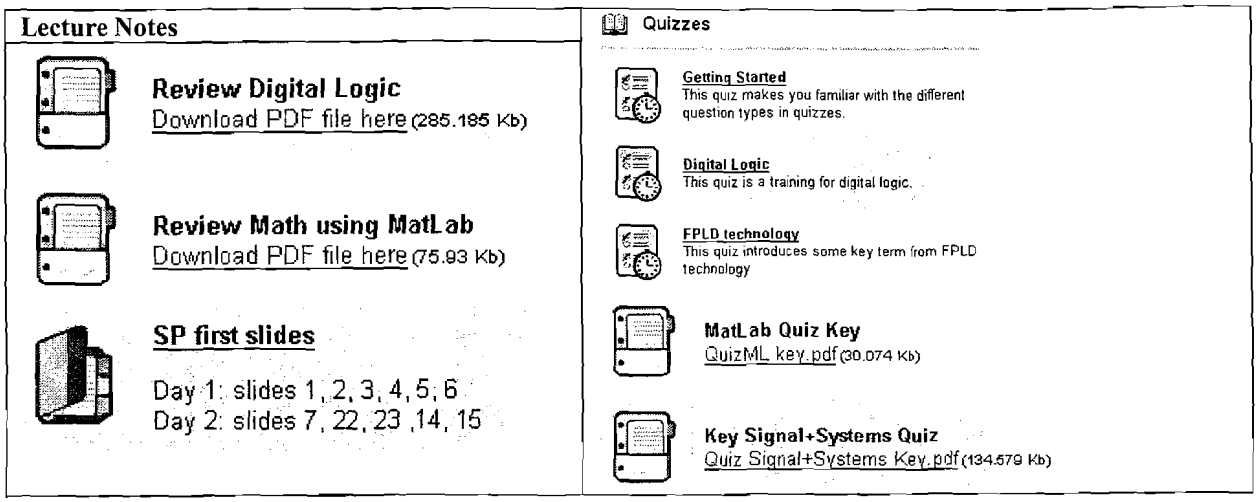
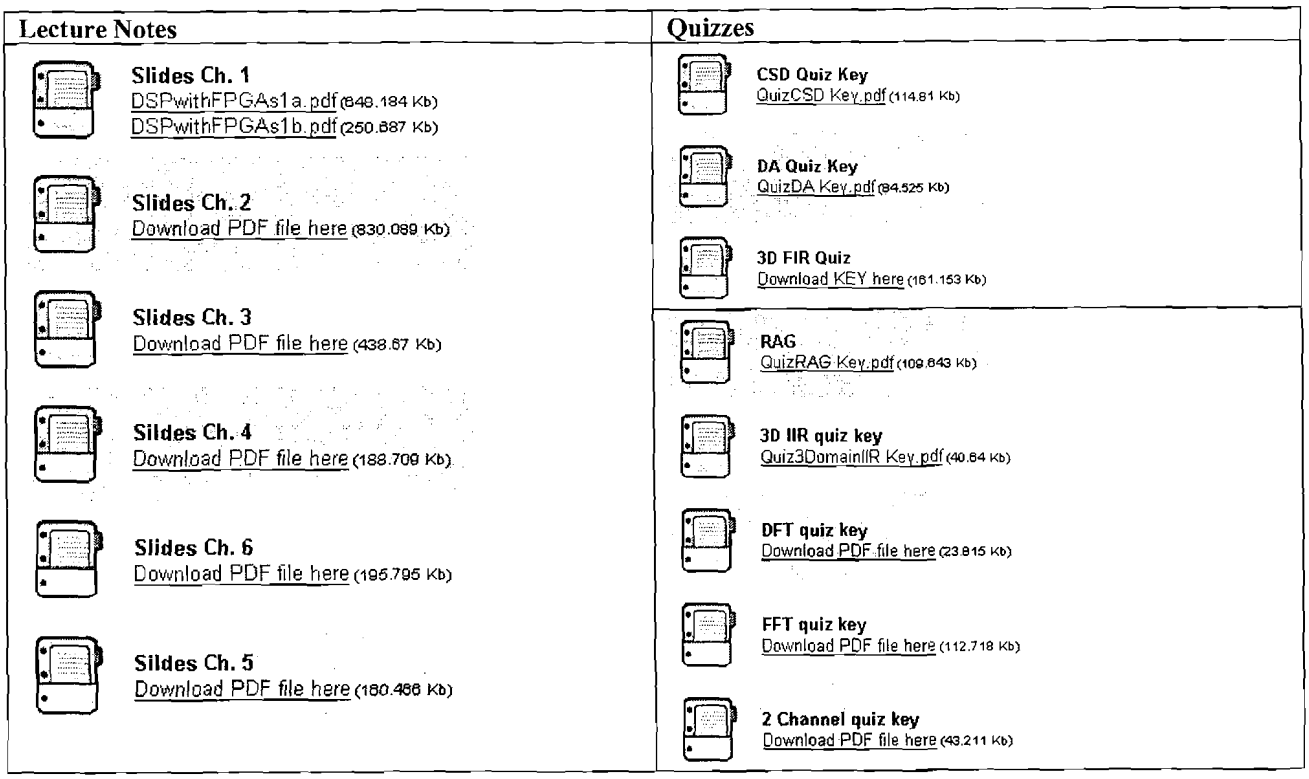


Figure 5: The course map.

Before we were able to use topics from the DSP with FPGAs[6] book, we reviewed digital logic, math, and linear algebra using MatLab, and signal and systems. For the signal and system review we used several sets of slides provided by the SP first book[7]. The following lecture notes and quizzes were used for the 3 reviews:

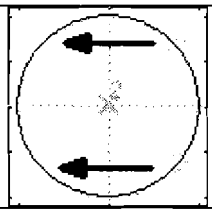


The first three quizzes listed are BB online quizzes, the last two show the answer keys to "pencil-and-paper" quizzes. In the remaining lectures we followed the topic selection given in our pervious paper[4] and Table 2. For each week we also developed online and paper quizzes that give the students immediate feedback on their learning progress. Notice that on Blackboard online quizzes, graphical entry is not available, and therefore most of the quizzes still had to be developed in a "paper-and-pencil" style. The developed lecture notes and quizzes are shown below:

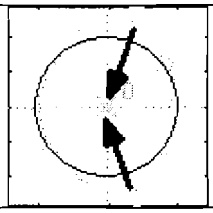


The midterm was scheduled during the week before spring break, and the topics to be included were the 3 review topics (from the start of the semester), labs, lecture notes, and quizzes until Chapter 3 on FIR filters. This leaves the topics of IIR, DFT, FFT and advanced topics for test 2, which can be conducted in the last week of class or during final exam week.

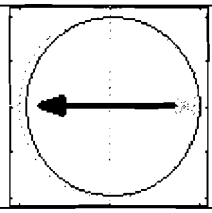
Besides the lectures notes and quizzes we have collected and recommended additional useful teaching material for the course. This includes 4 videos from SP first[7], 2 MatLab tutorial papers[8,9] found online, and a paper on A/D converters from a 11/2005 signal processing magazine and data sheet on devices and the Nexys board. Most of these files can be posted on the class Blackboard site under "Extra Documentation." Under the fair copyright act for teaching material we can use this material once per year without any additional permission from the copyright holder since posting on Blackboard is *not* considered to be the same as posting on the World Wide Web (WWW). Posting on the WWW would require copyright permission. There are many excellent short videos on the 3-domain representation (pole/zeros, transfer function, and time domain impulse response) of FIR and IIR filters in the SP first textbook. We have selected the following 4 videos as demonstrations of pole/zero movements:



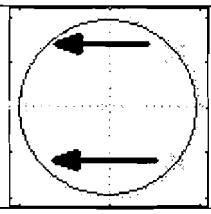
FIR filter with zero on the unit circle, i.e. bandstop.



FIR filter with zero moving across the unit circle, i.e. bandpass.



IIR filter with the pole moving on the real axis, i.e. lowpass and highpass.



IIR filter with the pole moving along the unit circle, i.e. bandpass.

Here is a listing of the "Extra Documentation" posted on the Blackboard site:

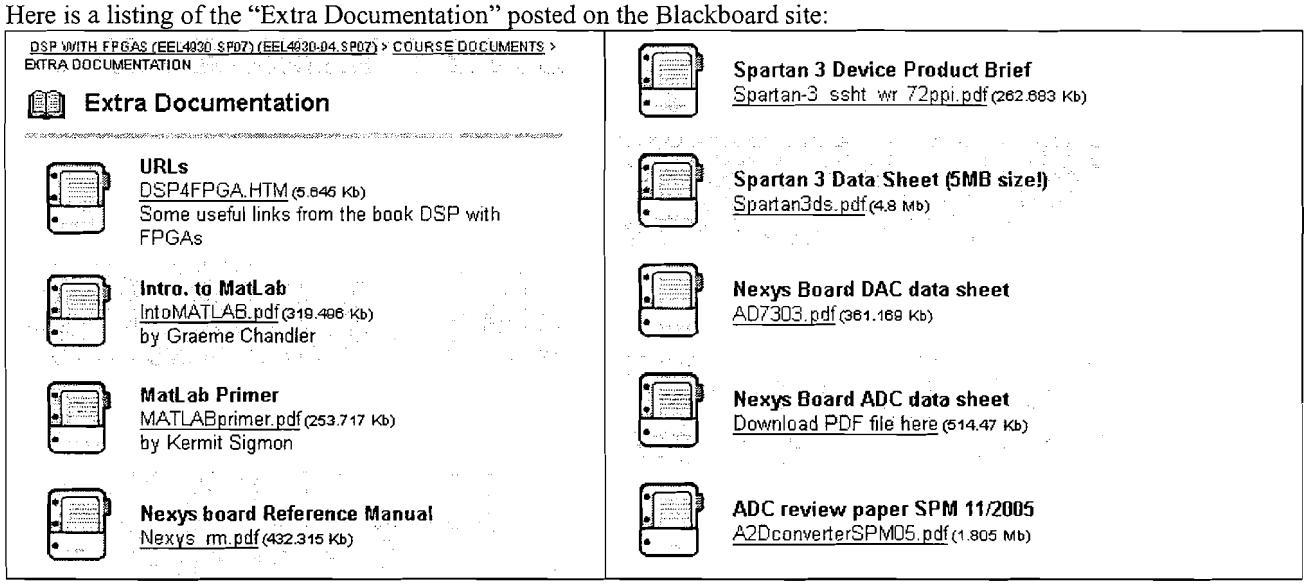


Table 1 shows the selected FPGA design examples used in the lecture notes from each chapter. We also include the VHDL synthesis results, since the Simulink design results are similar. Notice that in terms of speed, Simulink design results are often better, while in terms of area, the results are competitive, with similar required numbers of LUTs and embedded multipliers. The MatLab/Simulink design seemed to use more embedded multipliers while using less LUTs. The completion of all topics in the form of an "Introduction to DSP with FPGAs" textbook will then be included in a "full scale" EMD CCLI development.

Table 1: VHDL ↔ Simulink comparison for the Xilinx Nexys board.

Ch.	Design	Xilinx XC3S200-5ft256						Description
		VHDL			Simulink			
		LUTs	Mul.	MHz	LUTs	Mul.	MHz	
1	fun_text	32	0	215.7	32	0	162.7	Sine + triangular function generator
2	add_1p	48	0	156.0	31	0	167.0	32-bit pipelined adder with 1 stage
2	add_2p	63	0	143.4	31	0	173.4	32-bit pipelined adder with 2 stages
2	add_3p	88	0	152.5	47	0	190.0	32-bit pipelined adder with 3 stages
2	mul_ser	113	0	111.1	76	0	138.6	serial/parallel multiplier
3	fir_gen	57	4	86.4	57	4	86.8	Generic 4-tap FIR filter
3	fir_srg	118	0	49.7	50	0	74.1	4-tap FIR
3	fir_sym	64	0	87.8	37	0	86.8	4-tap FIR using symmetry
3	fir_csd	54	0	69.9	40	0	99.2	4-tap FIR using CSD coding
3	fir_tree	130	0	132.7	58	0	243.1	4-tap FIR using adder tree
3	fir_csd_sym	39	0	94.2	40	0	89.3	4-tap FIR using CSD and symmetry
3	fir_csd_sym_tree	51	0	181.4	32	0	239.5	4-tap FIR using all 3 optimizations
3	dasign	45	0	147.4	61	0	138.6	Distributed arithmetic FIR filter
3	dapara	18	0	104.2	55	0	104.1	DA full parallel FIR filter
4	iir	96	0	81.2	55	0	83.3	IIR filter first order
5	db4poly	168	0	86.7	115	2	118.7	Daubechies length-4 filter bank
6	ccmul	39	3	n/a	39	3	n/a	Complex multiplier
6	bfproc	71	3	49.9	103	3	40	Radix-2 butterfly processor
	Average	86.3	0.7	130.0	63.9	1.3	149.0	
	Gain %:				34.9	-50.0	14.6	

4. LABORATORY DEVELOPMENT

Based on the Altera labs developed in the first year, Xilinx lab experiments using the new Nexys board form Digilent were developed. We used a two-week lab cycle. All labs were conducted in a two-hour Friday afternoon session by the instructor, without a TA. On the first Friday the pre-lab and design-lab document were posted on Blackboard and given as a handout. The due date for the pre-lab was one week from posting, and the due date for the design was two weeks from posting. Because of this two-week schedule, there was always an overlap of the labs that allowed faster students to complete the labs early and the slower students to use the full two weeks for completion. Since the labs were graded by the instructor one by one, it was not necessary to post solutions, and the "deadlines" for the labs therefore didn't need to be "hard" as is the case when solutions are posted for quizzes.

The Nexys board in use includes a 200K Gate device from the Spartan 3 family of devices. The floorplan of the device is shown in Fig. 8. Since tutorial files for Nexys are only available for VHDL, we first developed a Nexys tutorial for the MatLab/Simulink flow and posted the following files on the WWW[12]:

Files	Size	Contents
Nexys_AD_DA.mdl	**102K**	Simulink design generating a sine wave and using all peripherals including A/D/ and D/A, LEDs, 7 segments, switches, and buttons
Nexys.ucf	**1K**	The port specification
nexys_ad_da_clk_wrapper.bit	**131K**	ISE compiled files ready for download
NexysDemo.MOV	**167,496K**	Quicktime movie with overview of design flow
Nexys.html	**18K**	Quicktime movie text

One particularly excellent lab setup is possible with a complete mobile environment using a Laptop, the Nexys board, and an LCD-scope by Velleman, as shown in Fig. 7. The laptop and scope are fully mobile. If we use the USB programming mode in Nexys, then the Nexys power is also supplied via the USB cable, and no additional power adapter is needed. The design flow in this setup consists of the following three steps:

1. Design the circuit in MatLab Simulink and generate the HDL files with System Generator.
2. Start the ISE software and load the project. Add the UCF file to the project and make a full compile by clicking on Generate Programming File near the bottom of the **processes** window so that the bit files are generated.
3. Use the Digilent ExPort tool to program the Nexys board.

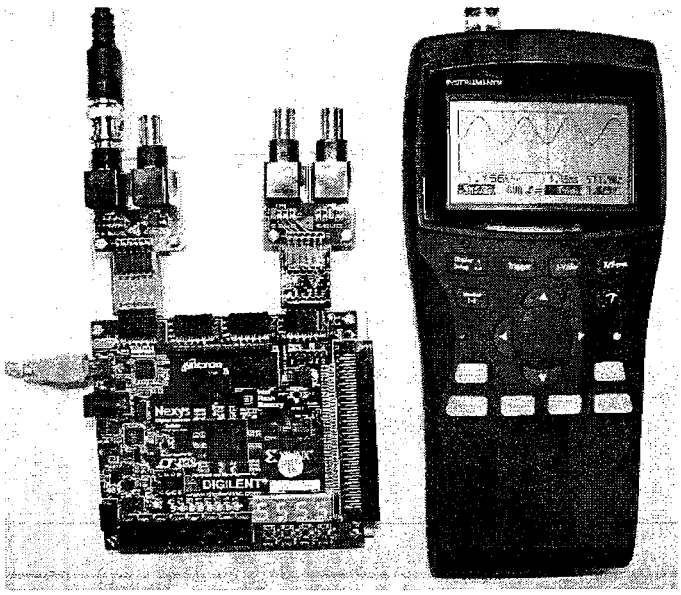


Figure 7: Full mobile lab environment: Nexys power supply via laptop USB and HPS10 Velleman scope.

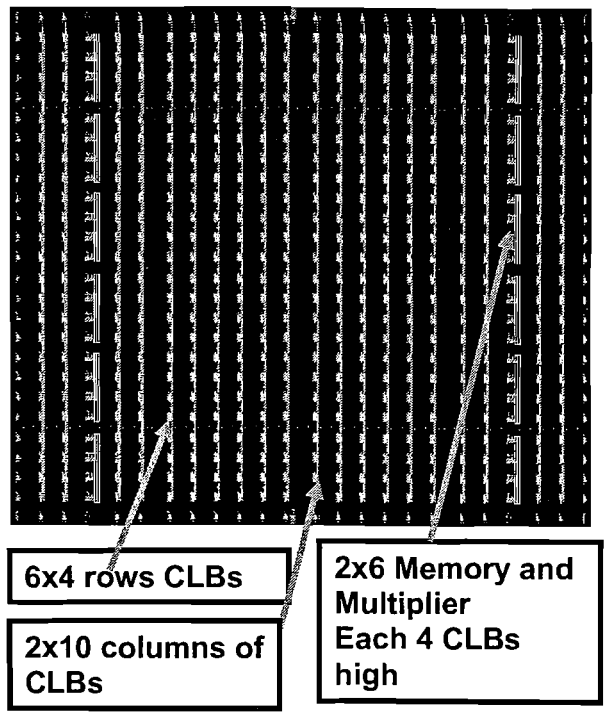


Figure 8: Xilinx Spartan 3 XC3S200 floorplan. One CLB has 4 slices with 2 LUTs each.

Although this design process is a little bit more labor intensive than the single push bottom in the Altera flow, the students did not have a problem getting the flow to work during our very first lab. The design data for the total number of LUTs, BlockRAMs, and embedded multipliers are available from the Σ View Design Summary in the ISE Design Summary. The registered performance timing data can be found by looking at the **Post Place and Route Static Timing Report,** under **Detailed Reports** (at the bottom of the design summary). These were the only instructions necessary for the students using the ISE software.

Students should also be encouraged to buy the MatLab student version of the software (which costs about $100) This allows the student to design the circuits at home. The student packages include a basic MatLab/Simulink version along with tutorial books[10,11] and some tool boxes. The Xilinx System Generator and ISE software can be downloaded for free from the Xilinx webpage.

Funding for 30 hours per week was available and was used to pay for the development of the Xilinx lab for 3 RAs. The RAs first had to learn the Simulink design flow and then develop a lab manual and pre-lab questionnaire for the experiments. With the support of the instructor, the RAs developed the pre-lab questionnaires. After successfully passing

the prelab questionnaires, students were allowed to conduct the laboratory experiments. The advantage of this approach is that the pre-lab quiz reduces the number of questions asked to the TAs and guarantees that the students are well prepared before entering the lab.

The following listing gives a brief overview of the developed experiments and the major files posted on the web[12].

Lab 1: Introduction to Simulink and DSP Builder

In this lab we introduce the Simulink environment for Digilent's Nexys Development board. In the **pre-lab** we first study important facts on the FPGA device in use and the Nexys board, try to find Simulink blocks in the tool boxes, and analyze accumulators in different bit widths. In the **design part** we first use a completed design to demonstrate the design flow. Then, in a second part, the students design the same circuit (i.e., a sine wave generator) using the library elements. An incomplete design is available that contains the serial/parallel converter clock generator used in the A/D and D/A conversion process. The files used in this lab are:

Lab Intr. to Simulink and Xilinx System Generator
lab1.mdl (106.902 Kb)
lab1.ucf (1.235 Kb)
lab1inc.mdl (96.096 Kb)
Lab Intro Simulink.pdf (337.067 Kb)

Lab 2: Number Systems and Quantization

This lab introduces fractional number systems. In the **pre-lab** we first review signed and unsigned number systems and their ranges. We then discuss a Xilinx `Fix_5_2` format, i.e., a fractional number with 3 integer and 2 fractional bits. In the **design part** the students design and simulate a 5-bit bus implementation with different numbers of fractional bits.

The files used in this lab are:

Lab 2: Number Systems and Quantization
Download instructions here (295.574 Kb)

Lab 3: Introduction to Signal Flow Graphs

In this lab Signal Flow Graphs (SFGs) or system analysis and system synthesis are introduced. DSP systems can be described by different methods. Most frequently, DSP systems are described by a difference equation, the z-transform, and the SFG. When designing a system, it is very important to analyze the system's properties mathematically, which is done by the three methods noted above in the **pre-lab**. In the **design part**, students develop and design three different systems (FIR, IIR, and one nonlinear) based on a given impulse response (for FIR and IIR) that have been studied in the pre-lab.

The files used in this lab are:

Lab 3 Signal Flow Graphs
Download instructions here (207.118 Kb)
sfg.mdl (28.701 Kb)

Lab 4: Introduction to MatLab M-File Scripts

In this lab we introduce MatLab M-file coding. MatLab is a powerful DSP interpreter that allows you to quickly and efficiently develop large function tables and test bench data for your FPGA design. We extend our function generator

from Lab 1 with additional test functions and also write M-file scripts to test a complex multiplier design that needs 3 real multiplications and 5 add operations. In the **pre-lab** we compute with "pencil-and-paper" the results we later expect in the design implementation. In the **design part** the students design a function generator for 4 different functions and a complex multiplier.

The files used in this lab are:

Lab 4 M-Files
funcgen.m (0.592 Kb)
Download instructions here (276.306 Kb)

Lab 5: INTRODUCTION TO FIR FILTERS

In this lab we introduce finite impulse response (FIR) filter design. Filters are one of the most important elements in DSP and are typically used to isolate a specific frequency band of a signal. FIR filters have low quantization sensitivity, simple structures, and can be easily designed to have a linear phase response. During the **pre-lab**, students analyze the behavior of a three tap moving average filter and develop a reduced adder graph. In the **design part**, a halfband filter using direct coefficient coding and a reduced adder graph is designed. The students are also asked to bring their own pair of headphones to the lab so they can hear the effects of noise and filtration.

The files used in this lab are:

Lab 5 FIR filter
F5direct.mdl (60.316 Kb)
F5transposed.mdl (49.778 Kb)
Download instructions here (604.687 Kb)
maf.mdl (44.351 Kb)
showfft.m (0.233 Kb)

Lab 6: INTRODUCTION TO IIR FILTERS

In this lab, students are introduced to the design of Infinite Impulse Response (IIR) filters. Filters are one of the most important elements in DSP. IIR as FIR filters are typically used to isolate a specific frequency band of a signal. IIR filters are of particular interest because, with just a few coefficients, relatively sharp transition bands can be realized. During the **pre-lab**, students compute the filter coefficient for a first-order system from given transition band specifications and compare properties of FIR and IIR filters. In the **design part**, they design a first-order IIR filter and a third-order direct form filter.

The files used in this lab are:

Lab 6 IIR Filter
IIRorder1.mdl (33.618 Kb)
IIRorder3.mdl (41.562 Kb)
Download instructions here (232.401 Kb)
setup_iir3.m (0.425 Kb)
showfft.m (0.229 Kb)

Lab 7: INTRODUCTION TO DISCRETE FOURIER TRANSFORM

In this lab, students are introduced to the design for a Discrete Fourier Transform (DFT) using the Goertzel iterative computation. The DFT is described by the following equation:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \qquad k = 0,1,.....N-1 \tag{1}$$

$$\text{with } W_N^{kn} = e^{-j2\pi kn/N} \tag{2}$$

The DFT is an important DSP tool and can be used not only to compute an approximation of the Fourier Transform but also to build narrow band filters without the need of sophisticated filter design tools. The Goertzel algorithm can be used to implement a single DFT component via a first-order IIR filter. During the **pre-lab**, the students work with "pencil-and-paper" test data to compute a length-8 Goertzel filter. In the **design part** an 8 point DFT using the Goertzel algorithm is designed.

The files used in this lab are:

Lab7 DFT
goertzel.m (0.34 Kb)
Goertzel.mdl (39.857 Kb)
Download instructions here (211.167 Kb)

Lab 8: Introduction to the Fast Fourier Transform

In this lab the Fast Fourier Transform (FFT) design is introduced. The FFT is one of the most important DSP objects and is used not only to compute an approximation of the Fourier Transform, but also to enable fast convolution, a very time consuming filtering operation when done in the time domain. In the **pre-lab** students develop the Butterfly graph for a length-8 FFT and compute with "pencil-and-paper" the FFT output test data for a triangular input signal. In the **design part**, the students design an 8-point radix-2 FFT using the principle of decimation in frequency.

The files used in this lab are:

Lab 8 FFT
Lab8.mdl (99.282 Kb)
Download Instructions here (183.75 Kb)

Most of the laboratory experiments started with an incomplete design that students had to complete, simulate, compile, and verify using FPGA hardware, see Figs. 9 and 10.

The laboratories were conducted in a lab with 12 work stations that included the boards, programming software, power supplies, function generators, and oscilloscopes. All work stations had complete installations of the necessary software. Due to space and financial reasons, hardware equipment was available for 4 stations at a time. But since the USB board programming mode and a Velleman scope were used, the hardware could be moved from one station to the next within minutes. For the Xilinx lab, we used a new Nexys board from Digilent, which allows a direct connection of A/D and

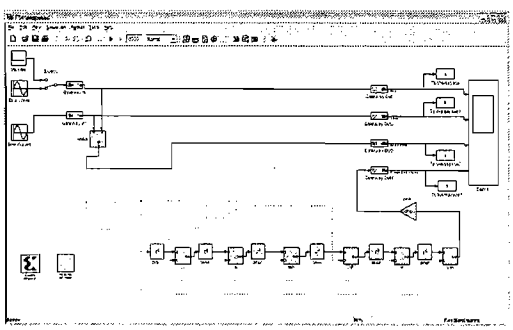


Figure 9: Incomplete RAG-n FIR lab

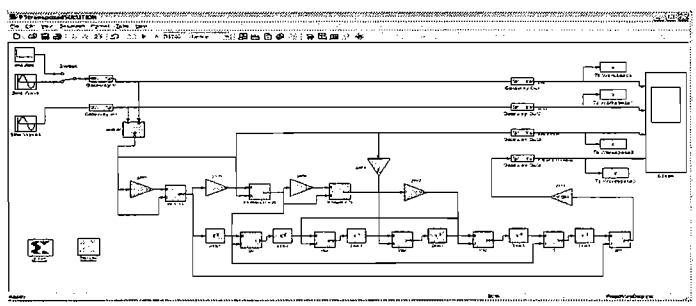


Figure 10: RAG-n FIR Lab key

D/A daughter boards to the I/O pins without the need for an additional adapter as in the previous Digilent S3 boards. However, the A/D and D/A converters have a serial interface and blocks, which need to be provided by the instructor for the students to communicate with the essential I/O units.

5. EDUCATIONAL MATERIAL EVALUATION

After the EMD was complete, we asked consultants from academia and the industry whether or not they thought the material was complete or if there were any improvements that possibly needed to be made. Since the instructors themselves tended to be more biased, we reported only the outside reviewers' feedback. For the Altera EMD, we received feedback from 3 people, and for the Xilinx EMD 4 responses. Table 2 shows the evaluations.

Table 2: EMD evaluation by Altera (3) and Xilinx (4) experts from industry (3) and academia (4).

Topic	Altera	Xilinx	Total
Logic Review	4.7	4.3	4.5
Matrix + Calculus	4.3	4.1	4.2
Signal + Systems	4.8	4.0	4.4
Overview DSP	4.5	4.9	4.7
FPGA Technology	4.0	4.5	4.3
Computer Arithmetic	4.5	4.5	4.5
FIR filter	4.7	4.5	4.6
IIR filter	4.3	4.0	4.2
Fourier Transform	4.7	5.0	4.8
Multirate DSP	4.5	4.0	4.3
Laboratory	5.0	4.8	4.9
Syllabus	5.0	4.3	4.6
Mid term	4.7	5.0	4.8
Final Exam	4.7	5.0	4.8
BB Webpage	5.0	5.0	5.0
Overall	4.62	4.52	4.57

Overall the evaluation was very encouraging. Small improvements like adding topics to IIR filters or using an IP core in the FFT laboratory will be considered in the next teaching cycle.

6. ACKNOWLEGEMENTS

The authors acknowledge NSF support under CCLI award DUE-0442600. The authors would like to thank Altera and Xilinx for the provided hardware and software under the university programs. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors. The help of the following consultants and program manager is acknowledged: B. Esposito, M. Phipps, R. Maroccia, (Altera) J. Weintraub, A. Acevedo, C. Sepulveda, C. Dick (Xilinx) P. Athanas, (Virgina Tech), H. Hamel (Raytheon), W. Moreno (USF), S. Connors (Boeing-SVS), J. Hallman (Harris Corp.), E. Manolakos (Northeastern University), E. Zurek (USF). We thank the following RAs: Andrew Hoover and Estella Kim for the Altera laboratory development, and Alejandro Canosa, Doug Murphy, and Stephanie Canosa for the Xilinx laboratory development. Products and company names used in this article may be trademarks of their respective owners.

7. CONCLUSION

We have presented results of an undergraduate course on DSP with FPGAs based on the Xilinx tool flow. Lecture notes, laboratory experiments, and quizzes were developed and evaluated for 2 days each by consultants from academia and industry. A first course was conduced in Spring 2007 and is currently ongoing (Spring 2008). A second Xilinx-based course is being presented at FAMU-FSU. The design examples selected for the lecture notes demonstrated that the Simulink design flow for FPGAs has become a viable design path. The Simulink results were competitive in size (LUTs and embedded multipliers) and better in speed than the hand optimized VHDL coding.

In conclusion, the Simulink design flow for FPGAs is an interesting alternative to introduce students at an early level to DSP hardware design concepts.

REFERENCES

[1] G. Moore, "Cramming More Components onto Integrated Circuits," Electronics Magazine 19, April 1965.

[2] Blackboard 6 Manual http://company.blackboard.com/docs/cp/learning_system/release6/instructor/

[3] Uwe Meyer-Baese, A. Vera, A. Meyer-Baese, M. Pattichis, R. Perry, "Discrete Wavelet Transform FPGA Design using MatLab/Simulink," Proc. SPIE Int. Soc. Opt. Eng., April 2006, Vol. 6247, pages 624703-1-10.

[4] Uwe Meyer-Baese, A. Vera, A. Meyer-Baese, M. Pattichis, R. Perry, "Smart Altera Firmware for DSP with FPGAs," Proc. SPIE Int. Soc. Opt. Eng., April 2007, Vol. 6576, pages 65760T1-12.

[5] B. Fawcett, "Synthesis for FPGAs: An Overview," Proceedings WESCON, Sep. 1994. p. 576-580.

[6] U. Meyer-Baese. "Digital Signal Processing with Field Programmable Gate Arrays," 778 pages 3. edition, Springer Press, Heidelberg, 2007.

[7] J. McClellan, R. Schafer, and M. Yoder, "Signal Processing First", *Prentice Hall, Upper Saddle River, NJ* 2003.

[8] Introduction to MatLab by Graeme Chandler, Mathematics Department, The University of Queensland

[9] MatLab Primer by Kermit Sigmon, Department of Mathematics, University of Florida

[10] The MathWorks Inc., "Learning MATLAB 7", Release 14, ISBN 0-9755787-0-9, *July* 2004.

[11] The MathWorks Inc., "Learning SIMULINK 6", Release 14, ISBN 0-9755787-3-1, *October* 2004.

[12] Uwe Meyer-Baese, online CCLI project page posting: `http://www.eng.fsu.edu/~umb`