AC 2012-5510: WORK IN PROGRESS - TEACHING HARDWARE IMPLEMENTATION OF DIGITAL SIGNAL PROCESSING ALGORITHMS ON FPGAS

Dr. Nader Rafla, Boise State University

Dr. Nader Rafla, P.E., received his M.S.E.E. and Ph.D. in Electrical Engineering from Case Western Reserve University, Cleveland, Ohio in 1984 and 1991 respectively. His Doctoral research concentrated on object recognition and localization from range image data, force-torque, and touch sensors data.

From 1991 to 1996, he was an Associate Professor in the department of Manufacturing Engineering at the Central State University. Where he taught courses related to the electrical engineering component of the program. In the mean time, he developed and was involved in a research program in applied image processing.

In January, 1997, He joined the newly developed electrical and computer engineering program at Boise State University where he is currently an Associate professor and chair of the Electrical Engineering Department. He led the development and starting of the M.S. of Computer Engineering; He taught several courses and supervised numerous M.S. thesis and Senior Design Projects. He also has conducted research and consulted in R&D for Micron Technology, Hewlett Packard and others.

Dr. Rafla's area of expertise is systems on a programmable chip and embedded & microprocessor-based system design; Neuromorphic systems; and implementation and hardware architectures of digital image and signal processing algorithms applied to recognition, identification, inspection, automation and control. He is a senior member of the IEEE organization and several societies and a member of the ASEE organization.

Work in Progress - Teaching Hardware Implementation of Digital Signal Processing Systems on FPGAs

Abstract

In today's world the amount of sampled data is increasing as more efficient and faster analog to digital converters (ADC) are becoming available. There is a need to implement Digital Signal Processing (DSP) solutions in hardware to accommodate the current processing rates of this data. Field Programmable Gate Array (FPGA) platforms offer enormous resources suitable for such implementation. In the mean time, they allow the hardware solutions to be prototyped, implemented, and tested with ease. However, in academia, there still exists a gap between developing systems using the traditional skills gained in DSP courses and implementing them on FPGAs. At Boise State University, the author had developed and taught a new graduate level course titled "Hardware Implementation of Digital Signal Processing" as part of the Masters and Ph.D. program in Electrical and Computer Engineering to close this gap. This paper discusses the topics chosen, course outline, practical exercises and the tools used in teaching this course. It also presents an assessment of the effectiveness of the developed course organization, practical exercises, and the teaching methodology. It also describes how traditional DSP concepts are blended into the implantation on FPGAs.

Introduction

Almost all Electrical Engineering curricula include a course in DSP theory and another in Digital Systems Design using Hardware Description Languages (HDL). Students usually struggle with integrating the knowledge gained in these two courses to develop and implement a working DSP-based system[1]. The author had developed a model for a new course to teach students the entire design process for DSP systems from specifications and simulation to implementation and verification. Furthermore, the course exposes students to advanced concepts in hardware design and DSP such as floating point arithmetic and the Coordinate Rotation Digital Computer (CORDIC) algorithm, Finite Impulse Response (FIR) filters, Infinite Impulse Response (IIR) filters, Systolic Filters, Multichannel Filters, Poly-phase filters, adaptive filters, and Fast Fourier Transform (FFT). The primary emphasis is on the implementation cost tradeoffs and timing issues involved in designing compact, efficient, and low-power hardware solutions for DSP systems. MATLAB/Simulink[2] are used for studying and understanding the DSP algorithms, Xilinx ISE suite of tools including System Generator[3] for design & synthesis of the hardware solutions into FPGAs, and QuestaSim[4] for function simulation & design verification. This course has practical exercises where students individually design, test & verify, and implement into an FPGA a series of small modules. Each exercise covers in detail all aspects of the implementation of a particular DSP concept. These exercises then climax into a major final project, proposed by the students, that requires the research, hardware design, simulation, and implementation of an advanced DSP system.

This paper describes the contents of the course “Hardware Implementation of Digital Signal Processing” taught at Boise State University. It explains the course goals, infrastructure and practical exercises along with the software tools that are widely used by industry.

Course Objectives

The purpose of this course is to provide students with the knowledge necessary for using FPGAs to efficiently implement DSP systems. It provides a strong foundation for the design techniques necessary to design DSP systems and implement them on FPGAs through lectures and practical exercises covering the many aspects of the design and implementation process, tools, and methodologies. Therefore, the objectives of the course are:

- To educate students the fundamental DSP concepts, algorithms and techniques for implementation on FPGAs;
- To familiarize students with the software design flow from a DSP algorithm to real time implementation on FPGAs using Matlab/Simulink and System generator;
- To teach students about the dataflow through FPGAs and how to use distributed memory, Block Random Access Memory (BRAM), registers, Shift Registers Lookup Tables (SRL), and DSP slices to properly implement DSP systems;
- To describe the advantages of using FPGAs over traditional processors for DSP designs;
- To practice the different implementation techniques of Digital Filters, and Fast Fourier Transforms (FFT) on FPGAs.

Course Outline

This course is taught over a period of fifteen weeks (three 50 minute periods per week). Each week consists of a lecture (one period) and one practical exercise (two periods). Students work individually on these exercises in class and continue at home then present, compare and discuss the results of their implementation techniques during the week following the completion of each exercise. In addition, there is a major final project proposed by the students. To achieve a passing grade, students must complete all practical exercises (70%) and finish and demo their project (30%). Two textbooks[5,6] are used in this course along with lecture notes developed by the author. The Xilinx user’s guide is also heavily used by the students. The course is divided into three parts as explained next.

The first part provides a review of the DSP theory, algorithms and design techniques. Since this is a necessary prerequisite for the course, each student prepares and presents a concept supported by an example.

The second part introduces the features of FPGA architecture pertained to DSP implementation and provides a tutorial on the entire design flow from a concept to simulation to implementation using an example design. The design flow usually starts with building the design in Simulink and verifying its functionality through simulation. Students then build the same design in System

Generator and practice with changing design parameters and sampling rate. Finally, the design can now be implemented on an FPGA development board.

The third part is a series of hands-on exercises. Students are given the opportunity to consolidate their learning through a series of interesting and informative DSP and digital communication design problems that incorporate the principal aspects of the Xilinx FPGA design flow. In this part, students explore the advantage of using FPGAs over traditional DSP processors through employing their flexibility and reconfigureability. They learn how to take advantage of the distributed structure of FPGAs in a number of ways. For example, arithmetic word lengths can be determined on an individual basis (unlike a typical DSP processor) thus permitting computational efficiency to be optimized. Furthermore, the degree of parallelism can be controlled to trade off the speed of a design against the area it occupies. The goal is to learn how to intelligently map DSP algorithms to FPGA hardware.

Course Practical Exercises

The practical design exercises given in this course illustrate several aspects of *DSP* for *FPGA* design. They include key concepts, architectures, applications, and the cost and performance metrics which must often be traded off by hardware designers. The suite of Xilinx software tools are used to support this process through all steps from design entry to implementation on a FPGA-based development board. The following is a brief description of each exercise.

Exercise 1: This exercise deals with the arithmetic operations required for DSPs. Implementation of both fixed and floating point operations on signed numbers are explored and the hardware cost of the different implementation techniques is examined. This exercise also exposes the students to several different methods for complex number implementation. The DSP application designed in this exercise is a four function generator and a multiplier. A screen shot of the multiplier is shown in Figure 1 and the implementation result using three different methodologies is given in Table 1 below.

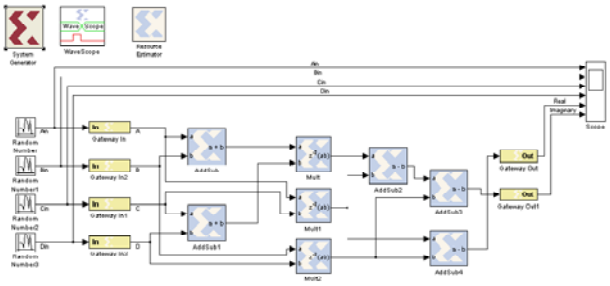

Figure 1: Design of a Complex Multiplier

Table1: Implementation Results of Complex Multiplier

	Distributed	Embedded Multiplier	DSP48
Occupied Slices	308	37	0
Slice LUTs	1206	144	0
LUT Flip-Flop Pairs	1222	144	0
DSP48	0	3	8
Minimum Period	4.015 ns	2.667 ns	2.667 ns
Maximum Frequency	249.066 MHz	374.953 MHz	374.953 MHz

Exercise 2: This exercise allows students to experiment with some of the issues around the implementation of digital Finite Impulse Response (FIR) filters. FIR filters are one of the essential elements of DSP, and are used for a wide variety of purposes. In this exercise students explore some of the possible constructions of these filters such as symmetric, systolic, multichannel, and polyphase. They also consider the performance and cost implications of using different weights and word lengths. Table 2 below shows the implementation results of four

different designs. The original design is using a direct decimation, the second is a polyphase that is further modified to produce the other two designs by dropping the zero coefficients from one phase and then the second phase.

Table2: Device Utilization Summary and Timing for Polyphase filters

Report	Result	Direct	Polyphase	Mod 1	Mod 2
Device Utilization Summary	Number of occupied slices	254	252	167	122
	Number of slice FFs	198	261	184	272
	Number of LUTs	663	644	340	264
	Number of bounded IOBs	57	54	53	52
Post Place & Route Stating Timing Report	Minimum Period	10.59ns	10.744ns	10.11ns	6.459ns
	Maximum Frequency	94.375MHz	93.075 MHz	98.912 MHz	154.823 MHz

Exercise 3: This exercise is similar to the previous one but for Infinite Impulse Response (IIR) filter. Students are exposed to design techniques specific to multiband filters and iteration free filters. Architectures for achieving high speed performance are explored utilizing pipelining through cutsets and dealing with the effect of rounding and quantization noise in hardware.

Exercise 4: This exercise leads the students in the process of implementing some of the commonly used adaptive signal processing algorithms, paying particular attention to FPGA design issues. The implications of feedback on the critical path of the design is also observed and studied. Examples of implemented adaptive filters are parallel adaptive Least Mean Square (LMS) filer, non-canonical LMS filter, and Square Root-Recursive Least Square (QR-RLS) filter.

Exercise 5: The fifth and last exercise sheds the light on the important issues related to the implementation of Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) algorithms. These algorithms are difficult and time-consuming to implement in hardware because of the large number of algorithmic and datapath options that must be considered in order to tailor the implementation to a particular application's cost and performance requirements. Topics discussed include: decimation in time and frequency; multi-path delay feedback; and delayed buffering.

Final projects are proposed by students. Examples of the proposed projects are: Numerically Controlled Oscillator (NCO); Cascaded Integrator Combo (CIC) filter; Channel Equalizer; Digital Communication Transmitter; Digital Communication Receiver; and Pulse Shaping.

Course Benefits and Assessment

This course is an important elective course to graduate students interested in the topics of DSP and reconfigurable hardware design. It plays a vital role in stimulating their interest to perform research in the area of hardware implementation of DSP systems. Through lectures, readings, and working with practical designs, students learn the pros and cons of different implementation methodologies. Each time the course is offered, its contents change to reflect the new trends in industry including any new features of the tools and hardware platforms.

After successful completion of the course, students reported that they had gained a well understanding of the techniques used in designing optimized DSP systems in hardware. The practical exercises taught them the best-practices used for DSP hardware implementations on FPGAs using state-of-the-art commercial tools and advanced platforms offered by modern FPGAs. A sample of the students' comments follows:

- This class interested me in hardware implementation of DSP algorithms.
- I took this course as an elective because I wanted to learn about how to actually implement DSP systems on FPGA.
- This course served as a bridge between the concepts I learned in the DSP course and my industrial knowledge of FPGA design and implementation.
- The class stimulated and provoked my thoughts about alternatives of filter design implementations on FPGAs.
- The practical exercises are carefully prepared and stimulated my interest to explore alternatives for better performance.
- The instructor gave me the opportunity to implement a design of my choice as a final project. I selected a design that I had been struggling with before taking this class. The knowledge gained during the class helped me to successfully implement it.

The course is offered during the Spring Semester of even years. It had been offered twice so far. 25% of the students are using these techniques on their Master's thesis while 10% are applying this knowledge into power control systems. The rest of students (65%) are working in industry and took the course out of their own interest to gain knowledge of the field to help them at work.

Conclusion

In conclusion, course contents and structure (lectures, practical exercises, and project) has met its goals. However, there exists a steep learning curve due to the diverse background required for it and it is a continuing struggle to achieve high level of quality while covering such difficult topics. It is rewarding to see students going over many stumbling blocks and reach the point of being able to develop and implement meaningful DSP systems. This course represents a good example for integrating the knowledge gained in two major areas of electrical engineering, digital signal processing and digital hardware design.

References

[1] S L Wood and S C Kemnitzer, *"First Year DSP Education in the Context of ECE Curriculum Reform,"* in Proc. IEEE 13th DSP and 5th SPE Workshop, Marco Island, FL, 4-7 Jan 2009, pp. 425 - 429.

[2] *"MATLAB/SIMULINK"* version 7. (R2008b). Natick, Massachusetts, The MathWorks Inc., 2008.

[3] "*ISE Design Suite and System Generator,"* Version 11.1, Xilinx Inc., 2009.

[4] *"QuestaSim Reference Manual,"* Version 6.3C, Mentor Graphics Corporation, 2009.

[5] Keshab K. Parhi, "*VLSI Digital Signal Processing Systems: Design and Implementation,*" Wiley, 1999, ISBN: 978-0471241867.

[6] Uwe Meyer-Baese, *"Digital Signal Processing with Field Programmable Gate Arrays,"* 3rd Edition, Springer, 2007, ISBN: 978-3540726128.