AC 2012-4159: INTRODUCING LABORATORIES WITH SOFT PROCESSOR CORES USING FPGAS INTO THE COMPUTER ENGINEERING CURRICULUM

Prof. David Henry Hoe, University of Texas, Tyler

David Hoe received his Ph.D. in electrical engineering from the University of Toronto. He held a position as a Staff Engineer at the General Electric Corporate Research and Development Center for five years prior to assuming his current position as an Assistant Professor in the Electrical Engineering Department at the University of Texas, Tyler, in 2008.

Introducing Multiple Soft Processor Cores Using FPGAs into the Computer Engineering Curriculum

Abstract

Soft processor cores are becoming an important component in state-of-the-art Systems-on-a-Programmable-Chip (SoPC) implementations. An SoPC design is a complete electronic system that is built on a reconfigurable integrated circuit, usually in the form of a Field Programmable Gate Array (FPGA). This paper will discuss the introduction of soft processor design into the courses within the Computer Engineering curriculum at the University of Texas at Tyler. Laboratories that utilize soft processor core design in our FPGA Design course and designs consisting of an array of soft processor cores to emulate multiprocessor designs in our Computer Architecture course will be described. Assessment in the form of project results, surveys, and instructor observation will be given.

Introduction

Continued advances in semiconductor technology over the past several decades have resulted in an exponential growth in the number transistors that can be fabricated on a single integrated circuit (IC). As a direct result of this, state-of-the-art Field Programmable Gate Arrays (FPGAs) can implement complex digital designs consisting of millions of logic gates at a speed comparable to custom integrated circuit designs but at a fraction of the development cost. Microprocessor implementations, known as soft processor cores because they are completely specified by a high level descriptor language, are now routinely included in FPGA-based designs. The capacity of the modern FPGA has reached the point where the implementation of an array of soft processor cores on a single chip is now feasible. Because these soft processor cores can be optimized by the designer for specific applications, lower power and improved speed compared with custom off-the shelf microprocessor-based designs are often attainable. Recent reconfigurable multiprocessor designs have shown the potential for improved performance. For example, some applications for these Multiprocessor-on-Programmable Chip implementations include a design for tracking multiple targets in an automotive application[1], a streaming multiprocessor design for bioinformatics processing[2], and a chip for routing packets in a networking application[3].

Since it is important to train our students in the latest technology used by practicing engineers in industry, this paper will discuss the introduction of soft processor design into the courses within the Computer Engineering curriculum at the University of Texas at Tyler. Laboratories that utilize soft processor core design in our FPGA Design course and the introduction of multiprocessor design using these soft processor cores into our Computer Architecture course will be described. This paper is outlined as follows. First, the relevant background information will be given, in terms of the educational context of the Computer Engineering curriculum, and the technical context concerning soft processor cores. This will include a discussion of the selection of the reconfigurable platform. The following sections will then discuss the development of the laboratories for the soft processor cores in the FPGA Design and Computer Architecture courses. The paper will conclude with a discussion of the results and look ahead to expansion of the laboratories in the near future.

Background on the Computer Engineering Curriculum

All electrical engineering students at our institution have the option of three technical elective courses, typically taken in their senior year. For those wishing to specialize in the Computer Engineering track, the Computer Architecture course is a required class, and the FPGA Design course can be taken as one of the electives. The Computer Architecture class focuses on the Instruction Set Architecture as the hardware/software interface with particular emphasis on the hardware implementation issues faced in the modern processor. The most recent course offering included a discussion of multicore processor design and the relevant software issues for interprocessor communication and parallelization of code.

The FPGA Design class introduces the students to reconfigurable logic and how to synthesize an FPGA design using VHDL. Circuit design issues in implementing FPGAs and applications for FPGA designs are also covered. The most recent course offering included soft processor cores, requiring the students to implement a basic design and to learn to program it in its assembly language code. A course project involving the design of a stop watch timer was required for the graduate students. This involved the use of the soft processor core to implement the basic functionality of a stop watch. The students were required to write VHDL code to interface the processor with the LED display and the push buttons on the FPGA development board and to write assembly code for the PicoBlaze processor.

Soft Processor Cores

This section provides a survey of soft processor cores currently available, discusses the rationale for the choice of the PicoBlaze soft processor core used in our laboratories, and provides some detail on the PicoBlaze processor core.

Xilinx and Altera represent the two companies that currently hold the greatest market share among FPGA implementations. Our laboratories are mostly equipped with FPGA development boards from Xilinx. The Basys 2 and Spartan-3E FPGA development boards are lower end boards that we primarily use for teaching purposes. Our labs are also equipped with several higher end (Virtex 5 and Virtex 6) boards that are used for research purposes. However, since all the Xilinx boards utilize the same synthesis software package (ISE software donated by Xilinx to universities) and the same high-level descriptor language (VHDL or Verilog) to specify designs, it is relatively easy for a student to migrate from the teaching to research oriented development boards.

Table 1 summaries some of the more popular choices that currently exist for processors implemented on FPGAs[4]. Focusing on the Xilinx FPGAs, the two main choices are between the MicroBlaze and PicoBlaze processors. The MicroBlaze processor is the more sophisticated of the two, featuring a 32-bit datapath, a Reduced-Instruction Set Computer (RISC) architecture, and separate memories for data and instructions. The processor can be easily implemented using Xilinx's Core Generator program. The MicroBlaze development package is also equipped with resources that allow for shared-bus and efficient point-to-point connections with other processors[5]. The PicoBlaze processor uses an 8-bit datapath, a small 64-byte data memory, 16 registers, a 1 K instruction PROM, and a single interrupt line. It is targeted for applications that require a simple 8-bit microcontroller. The basic architecture is shown in Fig. 1.

Table 1. FPGA-based Processors

Processor	Company	Bits	Comments
MicroBlaze	Xilinx	32	RISC – Core Generator
PowerPC	Xilinx	32	Hard processor core
PicoBlaze	Xilinx	8	Open-source VHDL
Nios II	Altera	32	Three types
Leon 3	Gaisler	32	RISC – Open source
OpenRisc 1200	OpenCores	32	RISC – Open source

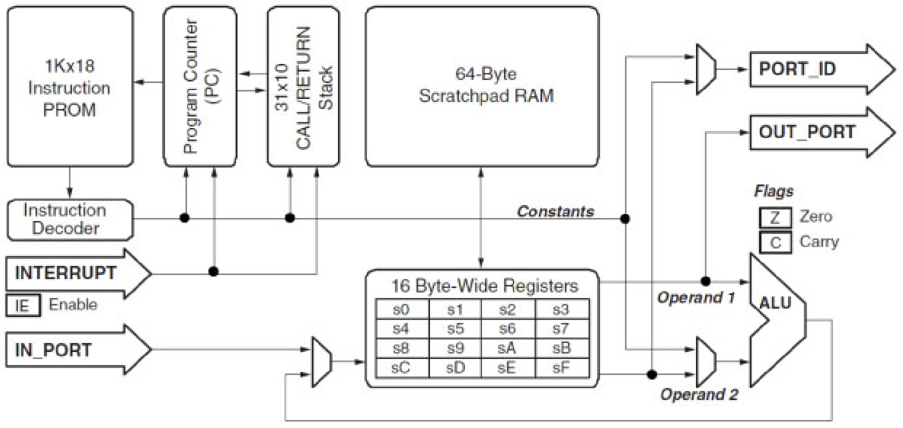

Figure 1. Architecture of the PicoBlaze Processor[6]

There are two advantages of using the MicroBlaze soft processor core. First, the use of the Core Generator makes it easier to implement and connect to other processors on an FPGA. Second, it uses a similar 32-bit RISC architecture that is described in the textbook used in our Computer Architecture class, the MIPS processor[7]. However, the decision was made to use the PicoBlaze processor in our labs for three reasons. First, with the PicoBlaze processor, one must build the interface logic for the ports to connect with other peripherals and processors and multiple interrupts must be specified separately using VHDL code by the user. It was deemed more beneficial for the student to be able to code the required interfaces rather than use a packaged solution. Second, the PicoBlaze processor is completely specified in open-source VHDL code, allowing it to be studied and changed by the student, while the MicroBlaze processor is implemented as an Intellectual Property (IP) core that is not accessible to the user. Third, the simplicity of the PicoBlaze architecture makes it easier to learn to program. As there is a limited amount of time in the course devoted to the soft processor core in both the FPGA Design and Computer Architecture courses, this was an important factor in our decision.

FPGA Design Laboratories

The FPGA Design class has a total of eight laboratory sessions integrated into the class, as summarized in Table 2. The first five labs give the students experience in writing VHDL code to implement simple logic designs, like adders and clock dividers, and to use the simulation and virtual logic analyzer tools for design verification and debug purposes. Some of the code needed to interface with the board is given to the students, such as the logic needed to drive the seven-segment display, but the students are required to analyze it and understand it for later use.

Table 2. Summary of FPGA Class Laboratories

Lab	Learning Objective
1. Half adder	Write VHDL code for a half adder and test its functionality
2. Full adder	Use the concept of hierarchy to implement a 1-bit full adder and test its functionality
3. Counter	Use a VHDL process to implement a counter and write a test bench to verify its functionality with a simulator
4. Clock Divider	Design a clock divider and use it feed a counter, display the results on the 4 digit display on the FPGA development board
5. Four bit adder	Learn to use the virtual logic analyzer tool (ChipScope) to test a four-bit adder design
6. Introduction to PicoBlaze	Learn to use the tools to write assembly code and to generate the ROM instruction file for the PicoBlaze processor
7. PicoBlaze Adder	Write an assembly language program to add two numbers, get the input from switches and output the sum to the display
8. PicoBlaze Counter	Understand how the basic interrupt mechanism works and use it to implement a counter with the PicoBlaze processor

The final three laboratories are devoted to learning how to implement and program the PicoBlaze processor. The students learn to use the additional tools required to assemble and debug the PicoBlaze assembler and to generate the instruction ROM file. The complete VHDL code for the PicoBlaze processor is supplied for these labs. The students need to write some of the support logic to interface the PicoBlaze processor with the I/O on the development board and to handle the interrupts in the final lab. At this point, the graduate students are required to implement the stop watch timer project with the PicoBlaze and to demonstrate its functionality. They are required to write VHDL code for the support logic and to use interrupts.

Computer Architecture Laboratory

For this course, only one laboratory session is devoted to building and implementing the multicore PicoBlaze array. The challenge of introducing this lab into the Computer Architecture class is that the FPGA Design class is not a pre-requisite, so not all students will have had the background in FPGA design and the concept of the soft processor core. The students are given a basic introduction to VHDL and this is used in the lectures to show how the various hardware components of the processor are specified.

For this course, the emphasis in the laboratory is to make the students appreciate one of the key challenges of multiprocessor design: implementing an efficient means of interprocessor communication. The laboratory reinforces some concepts for multiprocessor design introduced into the course. For example, the latest edition of the textbook of Patterson and Hennesy[7] introduces the concept of synchronization between processors when reading from a shared memory location (i.e., the implementation of lock and unlock instructions to give the ability to atomically read and modify a memory location). For reconfigurable processors, there are three methods proposed for communication between processors[4]. The first is direct connection, known as point-to-point. This method is the simplest and most effective for a small number of connections but is not area efficient when a large number of processors communicate with each other. Second is the traditional shared-bus approach which is used in uni-processor designs. It is inefficient for large systems since only one processor can use the bus at a time. A third option has been introduced recently, the idea of implementing a network-on-chip (NoC) approach[8]. This method borrows ideas from the networking of computers and applies it to an array of processors by implementing a small router on the FPGA to handle interprocessor communication on a single chip[9].

The lab focuses on the first method, the point-to-point connection. For this option, a mail box approach is used where a First-In-First-Out (FIFO) buffer is implemented between each processor to minimize the wait times for the cores that process data at different speeds. The lab involves implementing five PicoBlaze cores on a Spartan 3E FPGA. Four of the PicoBlaze cores generate a sequence of numbers that are offset by four (e.g., PicoBlaze 1 generates 0, 4, 8, …, PicoBlaze2 generates 1, 5, 9, …, etc.). The fifth PicoBlaze processor receives the sequence of numbers from the four other processor cores by polling them in a round-robin fashion through the FIFO interface. A schematic of the implementation is shown in Fig. 2. Each processor core consists of a PicoBlaze core with four FIFOs for receiving the input and four latches for holding the output. The code running on the processor labeled PicoBlaze 4 reads the number sequence from units 1, 2, 3, and 5 and assembles it into a linear count and displays the output to verify that the correct values have been received. A delay loop is included in the code to slow the count down so that it is discernible on the display.

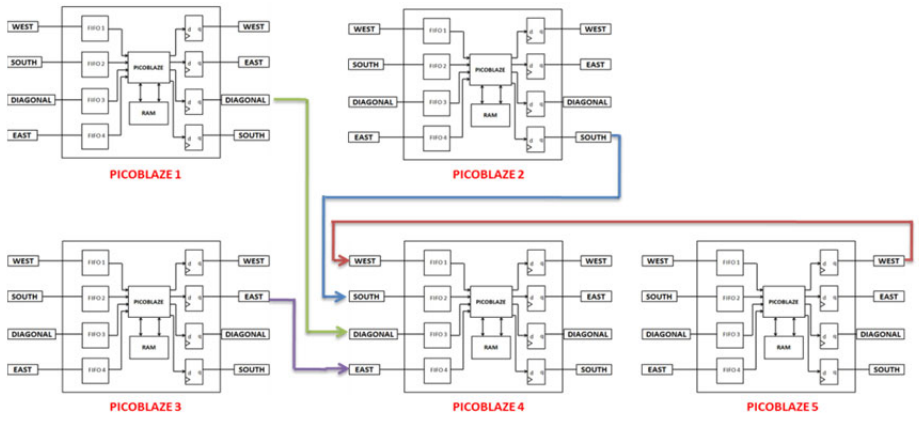

Figure 2. Array of five PicoBlaze processors where PicoBlaze 4 is the destination.

As the students may have a limited background on FPGAs for this class, a fairly detailed procedure is outlined in the lab for implementing the processor array. The instruction ROM for the four cores that generate the number count and send it to the one receiving core is given. As well, the code for the PicoBlaze and the FIFO buffers is provided. The students are required to generate the instruction ROM for the single PicoBlaze core that polls the other four processors for the numbers that they generate. The focus is on writing the assembly code to read from the FIFO. The pseudo code is given below.

Write to FIFO	Read from FIFO
read full_flag ;	read empty_flag ;
while full_flag = = 01	while empty_flag = = 01
read full_flag ;	read empty_flag ;
write data ;	read data ;

Figure 3. Pseudo code for the FIFO buffer write and read cases

Assessment

For the FPGA class, the ability of the students to comprehend and utilize the soft processor core was evaluated by their projects. The students were given an oral interview by the instructor, and they had to submit a final project report. The students were allowed to work in groups of two. One group managed to get the project working perfectly, five groups completed the project but with some aspect not working perfectly, and one group could not complete the project. The instructor's assessment for this first attempt at introducing soft processor cores into the course is that more labs need to be added. For the students that did not manage to get everything working properly, the ability to understand how to write the interface logic to handle the interrupts was one common difficulty. More time was needed to understand this. Overall, though, this first attempt can be deemed a success. The students grasped the basic concepts and understood the importance of using soft processor cores versus dedicated microcontrollers.

For the computer architecture class, the students were given a survey at the completion of the one lab on multiprocessors using the soft processor cores. Table 3 summarizes their responses with regard to their degree of comprehension. The class was composed of electrical engineering (EE) and computer science (CS) majors. Overall, the students felt they had a good grasp of the overall concepts for this laboratory.

A second set of questions surveyed the student attitudes towards learning more about embedded processor design and implementation of multiprocessor systems on FPGAs. Overall, it appears the students seemed favorable towards this goal. As expected, the CS students generally were not as interested in the hardware aspects, but did seem to feel they had a good grasp of the hardware concepts that were introduced in the lab. The student responses are summarized in Table 4.

Table 3. Student Understanding of the Multiprocessor Laboratory
(*Scale*: 1= Low, 2= Medium-Low, 3 = Medium, 4= Medium-High, 5 = High)

Statement	EE	CS
1. Understand the basic design flow for implementing a PicoBlaze processor on an FPGA.	4.00	4.00
2. Understand communication between processors using First In First Out (FIFO) buffers	4.00	4.33
Sample size	7	6

Table 4. Student Attitudes towards Reconfigurable Multiprocessors
(*Scale*: 1= Strongly Disagree, 2= Disagree, 3 = Neutral, 4= Agree, 5 = Strongly Agree)

Statement	EE	CS
1. I would like to learn more about embedded processor design using FPGAs.	4.14	3.67
2. The lab helped me appreciate the issues involved in interprocessor communication	4.14	4.17
3. There should be more labs of this nature in this class	4.14	4.00
4. I would like see an elective/graduate class devoted to embedded processors and multicore processor design	4.42	3.33
Sample size	7	6

Conclusions

With the continued advances in FPGA technology, more designers will be taking advantage of the flexibility and performance advantages obtained with designs that use soft processor cores. Indeed, as the number of logic gates and memory resources increase on state-of-the-art FPGAs, the ability to integrate multiple processor cores on an FPGA will grow accordingly. This remains an active area of research and development. This paper has described the introduction of soft processor cores into the Computer Engineering curriculum at our institution in keeping with an important trend in electronic and computer systems design. Initial results from the introduction of soft processor cores into our laboratories is encouraging. In upcoming classes, we intend to expand the labs in terms of quantity and sophistication. In particular, the other options for interprocessor communication, shared memory and network-on-a-chip concepts, will be added in upcoming course offerings. Eventually, a separate course offering on embedded processor design using FPGAs will need to be added to the curriculum.

References

1. J. Khan, S. Niar, A. Menhaj, Y. Elhillali, and J. L. Dekeyser, "An MPSoC architecture for the multiple target tracking application in driver assistant system," in *Proceedings of the 19th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '08)*, pp. 126–131, July 2008.
2. R. K. Karanam, A. Ravindran, and A. Mukherjee, "A stream chip multiprocessor for bioinformatics," *SIGARCH Computer Architecture News*, vol. 36, no. 2, pp. 2–9, 2008.
3. K. Ravindran, N. Satish, Y. Jin, and K. Keutzer, "An FPGA based soft multiprocessor system for IPV4 packet forwarding," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '05)*, pp. 487–492, August 2005.
4. T. Dorta *et al.*, "Reconfigurable multiprocessor systems: a review," *International Journal of Reconfigurable Computing*, 2010, 10 pages.
5. P. Huerta, *et al.*, "A Microblaze based multiprocessor SoC," *WSEAS Transactions on Circuits and Systems*, vol. 4, no. 5, pp. 423–430, 2005.
6. Xilinx, PicoBlaze 8-bit Embedded Microcontroller User Guide. Available at: http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf, 2011
7. D. A. Patterson and J. L. Hennessy, *Computer Organization and Design*. Morgan Kaufmann, Fourth Edition, 2009.
8. W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," *Design Automation Conference Proceedings*, pp. 684- 689, 2001.
9. H. C. Freitas, *et al.*, "Evaluating Network-on-Chip for Homogeneous Embedded Multiprocessors in FPGAs," *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007*, pp. 3776-3779, 27-30 May 2007.