

**2006-275: ON THE USE OF A SOFT PROCESSOR CORE IN COMPUTER
ENGINEERING EDUCATION**

Sin Ming Loo, Boise State University

On the Use of a Soft Processor Core in Computer Engineering Education

Abstract

The microprocessor course at most universities has traditionally been taught using a discrete microprocessor such as the Motorola 6800 series, Intel x86 series, or IBM PowerPC series. With the continued increase of usable field-programmable gate arrays (FPGA) gates and improvement of off-the-shelf soft processor core computer-aided design (CAD) tools, this practice is beginning to change. It is now possible to teach a microprocessor course using a soft processor core such as Xilinx Microblaze or Altera Nios. Such tools provide the user a graphical user interface to configure a 32-bit processor with the desired peripherals. This type of tool provides flexibility that was previously non-existent.

In the traditional method, interface devices need to be built into the microprocessor prototyping printed-circuit board. Otherwise, the additional interface devices will need to be built and wired to the main board. With the use of a soft processor core and an FPGA board with a sizable FPGA, all of the required interface logic can be built into the FPGA. Such hardware logic can be described using hardware description languages (Verilog or VHDL) or schematic capture. The FPGA functionality can be changed easily by downloading a new bit file (or configuration file) to the FPGA after re-running the synthesis tools.

These off-the-shelf tools have also been packaged with software development tools such as compilers, debuggers, and instruction set simulators, which are usually GCC-based. The encapsulation of hardware and software components into one FPGA design process also allows the hardware and software designs to be simulated – together – using a hardware description language simulator. However, the use of such tools doesn't come free (remember, there is no such thing as a free lunch!). There are many pitfalls when using a soft processor core in teaching. These off-the-shelf CAD tools usually have steep learning curves. In some cases, a non-working design can have problems in the hardware or software portion, which can be difficult to track. This paper describes teaching microprocessor design using a soft processor core, our experiences, our methodology, and the pitfalls in depth.

1 Introduction

During the last three decades, the microprocessor course has traditionally been taught using a discrete microprocessor such as the Motorola 6800 series, Intel x86 series, or IBM PowerPC series. The usual topics include the architecture of a selected microprocessor or microcontroller, assembly and C programming, and devices interfacing. Usually, an off-the-shelf prototyping board with the desired microprocessor/microcontroller is used in the laboratory to introduce the hands-on experience. This well-thought-out course structure has been working really well, and students completing this course usually have the skills to build a small-scale system.

Things are starting to change in embedded system design due to field programmable devices. In the old days, programmable devices were used as glue logic, but their use is no longer limited to this role. Programmable devices have also benefited from the shrinking of transistors, allowing more resources to be packed into a programmable device. With the continued increase of usable FPGA gates and improvement of off-the-shelf soft processor core computer-aided design (CAD) tools, it is now possible to teach a microprocessor course using a soft processor core such as Xilinx Microblaze [1] or Altera Nios [2]. Such tools provide the user a graphical user interface to configure a 32-bit processor with the desired peripherals. This type of tool provides flexibility that was previously non-existent.

In order to use the processor effectively and the FPGA efficiently, vendors package a suite of tools with software development tools such as compilers, debuggers, and instruction set simulators, which are

usually GCC-based. The encapsulation of hardware and software components into one FPGA design process also allows the hardware and software designs to be simulated – together – using a hardware description language simulator. In the field of system design, this is called hardware/software codesign. The goal of hardware/software codesign is to exploit the synergism that is possible when both the hardware and software portions of the design are performed concurrently and co-operatively. Hardware/Software codesign techniques have been motivated by the need to balance the competing actions of placing functionality in hardware to enhance performance or placing the same functionality in software to improve the degree of flexibility.

However, the use of such tools doesn't come free (remember, there is no such thing as a free lunch!). There are many pitfalls when using soft processor core in teaching. One of the pitfalls is the number of CAD software packages required for this methodology to work. These off-the-shelf CAD tools usually have steep learning curves. In some cases, a non-working design can have problems in the hardware or software portion, which can be difficult to track. This paper describes our experiences and our methodology in depth. The next section discusses the soft processor core teaching paradigm. Section three describes available off-the-shelf resources. Section four describes our experience from our small-scale deployment of such tools. The most important section of this paper is section five where the pitfalls of such technology are discussed. Our conclusions and future expansion are then discussed.

2 Soft Processor Core Paradigm

The two most popular soft processor core FPGA-based CAD tools in the market at the writing of this paper are Altera Nios [2] and Xilinx Microblaze [1]. Both tools are based on 32-bit RISC microprocessor with Harvard architecture. For the rest of the paper, all features mentioned are referring to both soft processor cores, unless otherwise specified. The processor core tools come with an assembly and a C cross compiler. No additional tools compilers are required for this to work. The vendor usually packages the processor core development tool in with their FPGA development tool. The FPGA development tool is packaged with HDL synthesis and implementation tools. The processor core development tool utilizes the FPGA development tools to synthesis a processor that can be used.

In order to do any programming or interfacing, the very first thing that one needs to do is configure a processor with the desired peripherals. These tools are usually packaged with some very popular peripherals such as UART, memory controllers, and JTAG debugging interface. Once a configuration is designed, the configuration will need to be synthesized and the configuration file can be used to configure the FPGA. By downloading this configuration to the FPGA, it will turn the FPGA into a microprocessor development platform. There are two methods by which code can be downloaded to the processor in the FPGA for execution. The first method is that when the software (refers to C or assembly) has been compiled, the software is inserted into the memory of the FPGA configuration file (using a script provided by the tool). With this method, changing the software will require the software to be re-inserted into the configuration file and re-download to FPGA. The second method is downloading the software after the FPGA has been programmed. The processor can be programmed by downloading the software binary. With this method, the debugging (which is JTAG-based) port will need to be configured into the processor.

Once the basic configuration set-up is completed, most of the traditional microprocessor laboratory exercises can be migrated to this environment. However, this paradigm requires none of the external discrete interface components in the traditional paradigm. All the necessary interface logic can be implemented using HDLs (Verilog or VHDL) or a schematics capture, component re-use, or even using a component that has been purchased or donated. These cores (or components) will need to be connected to one of the peripheral buses in the processor core so that software can be written to control them. The insertion of component (usually HDL based) to the processor core can be a complicated process and

sometime takes up more time than seems necessary (this process can be made explicit by having a tutorial to show only the necessary steps to completing the hardware core insertion).

With this approach, the students can be introduced to microprocessor programming as soon as the processor core is configured and downloaded to FPGA. Programming in assembly or C can be taught in the traditional method. There are many steps in this configuration process that seem incomprehensible at the early stage of this paradigm. One needs to be patient as these features and steps can be explained at a later time. With these tools and FPGAs representing the latest and greatest in the engineering world, one question an educator might have is how much will it cost?

3 Available Resources

Altera [3] and Xilinx [4] have university programs that are ready to provide many of the tools and initial hardware at no cost. This is usually to their advantage since what the students have been using in classroom will be carried over to their first job. In fact, if one has to pay for the se-up of the sixteen stations, it will cost about \$2k (one-time startup cost). The two most cost-effective FPGA prototyping kits suitable for processor core teaching are the Xilinx Spartan 3 starter kit [5] and the Altera Cyclone kit [6]. There are many learning resources available at tool vendors' websites. Software patches are frequently provided. A responsible instructor will need to make sure all of the CAD tools are patched with most updated service packs and updates. This will usually reduce many software problems. Altera and Xilinx also have good and useful online support databases. Knowing the right keyword to search usually solves most problems rather quickly. In addition, there are the Nios processor forum and the Microblaze processor forum. Both forums are good places to search for solutions and ideas. A list of tutorials and documents in PDF format for use in learning and teaching using soft processor core is provided. These documents, along with other application notes, can be found at both vendor's processor core webpage.

Altera Nios Resources [2]

- Nios Tutorial
- Nios Processor Architecture
- Nios Software Programming Manual
- Peripheral Interface Tutorial

Xilinx Microblaze Resources [1]

- Microblaze Tutorial
- Microblaze Processor Architecture
- Microblaze Software Programming Manual
- Peripheral Interface Tutorial

4 Our Experience

We have experimented with the deployment of soft processor core in a small scale in our digital systems rapid prototyping course and hardware/software codesign course. It is noted that since the students in this small scale study have been exposed to programming in assembly, C, and at least one processor architecture, the Nios or Microblaze processor architecture introduction is skipped and the tutorials are provided to the student to work at his or her own pace and to be completed within the given time. In the digital systems rapid prototyping course, the students were provided with a Xilinx Microblaze tutorial and a Spartan-3 starter kit [7]. A compatible Altera prototyping kit is also available [8]. The purpose of the tutorial was learning the basic steps to a basic processor core configuration for C code to be written. This short and quick introduction to Xilinx Microblaze was received very positively. Students were excited to be able to implement some of the functions that can be implemented quickly and easily using C. Another tutorial was provided [7] to learn how to interface the externally created hard-core to the processor.

In our hardware/software codesign course, two teams were introduced to Altera Nios and Xilinx Microblaze, respectively. With this introduction, the students proposed projects using Nios and Microblaze. Two projects were completed: 1) co-design of USB implementation using Nios, and 2) a music player using Microblaze. Both projects were implemented with co-design and completed as proposed. Both projects utilized hardware written externally and interfaced to the processor successfully.

Our small-scale usage of these tools provided us the confidence to offer a microprocessor course as a special topic in the very near future before incorporating it into our normal offering. During the same process, we have discovered many pitfalls to this approach of microprocessor teaching.

5 Pitfalls

The first problem with the use of this paradigm is the CAD tools. There are just too many CAD tools that will be involved in this paradigm and they change or are updated too quickly! Usually, multiple tools are integrated into one giant suite of tools. The encapsulation of multiple tools usually introduces many software bugs. This could potentially slow down the learning process or make the learning process much more complicated and difficult, which can be a big distraction. The potential solution is that the instructor must be very knowledgeable with the tools. But this is a kind of chicken and egg problem because the instructor may not have time to learn these new tools along with his or her many other responsibilities. One possible solution is to continuously train graduate students to be familiar with these tools and to be teaching assistants during laboratory and hands-on sessions.

The tools used in the soft processor core design environment are industrial strength tools with steep learning curves. Hours will be needed to get something simple started, but once that is completed many more things can be completed much more quickly. It is just that the early start-up overhead is very high. We have been approaching this learning curve problem by painstakingly designing multiple step-by-step tutorials (posted on the course webpage) so that students can learn the basic features on their own. This also provides materials for students to fall back on until they are very familiar with the basic features. In the introductory tutorial, we introduced the students to the most basic configuration so that the students can learn how to write interface code as soon as possible. Another tutorial is used to introduce to students to incorporate their hardware implementation into the soft processor core. There are many tutorials delivered with the CAD tools. These are usually very good for introducing the basic features of the tools. In addition, vendors have prepared many application notes with step-by-step and sample codes that are usually a good starting point to be used in the classroom.

The ability to change the hardware to improve performance and corresponding software to utilize the features is wonderful. However, when problems arise during the design process, it is very difficult to determine whether the problem is in the hardware, software, or the interface logic. Discovering this can be time-consuming and counterproductive. A simulator capable of simulating the processor with the desired software code running simultaneously with the hardware will be the appropriate tool for this occasion. Modelsim is one of those tools which allows the processor model and hardware to be simulated together. However, such tools are usually out of reach because of budget limitations or unacceptable simulation times.

Another problem is that vendors update the design software often. This means that what works in current the version of the processor core doesn't mean that it will work in the next release. This has been a problem for several years, and the situation will not get better in coming years. If you plan to use these tools for microprocessor instruction, consider yourself warned! Be very careful. One thing that you can do is develop the material and use it for at least one year before updating the tools to the latest version. Usually, vendors have major tool updates every twelve months or so with minor patches in the middle. Thus, you can stay with the stable version for a while.

The documentation always seems to fall behind! We have found many cases where the documented feature does not match the actual implementation. For example, we have found that the documentation describes a location to change for the external memory mapping. However, this didn't work because there is another location required changing and it is not being described in the documentation.

6 Conclusions

The use of processor cores to teach a microprocessor curriculum will be difficult at the beginning because of new hardware environments and involved multiple software tools. Our small-scale experiments include a processor core tutorial to put together a basic processor configuration, add external hardware to processor core, and interface to external devices. If handled correctly (tutorials for the students to learn and an instructor knowledgeable in those tools), it can be more beneficial than the traditional method. We have put in place a plan to develop one-semester microprocessor course using the Altera Nios DE2 development kit [9].

Acknowledgement

Our experiences wouldn't be possible without hardware and software donations from Altera and Xilinx. These donations are greatly appreciated. Aldec has also provided software and hardware simulators that allow a complete Microblaze system to be simulated.

References

- [1] Xilinx Microblaze Soft Processor Core, <http://www.xilinx.com/microblaze>, visited: 12/21/2005
- [2] Altera Nios Soft Processor Core, <http://www.altera.com/nios>, visited: 12/31/2005
- [3] Altera University Program Webpage, <http://www.altera.com/universityprogram>, visited: 1/9/2006
- [4] Xilinx University Program Webpage, <http://www.xilinx.com/univ>, visited: 1/9/2006
- [5] Digilent Spartan-3 Starter Kit, <http://www.digilentinc.com>, visited: 12/31/2005
- [6] Altera Cyclone Development Kit, <http://www.altera.com/universityprogram>, visited: 12/31/2005
- [7] Xilinx Microblaze Tutorial, http://www.xilinx.com/support/techsup/tutorials/edk_tutorials.htm, visited: 12/31/2005
- [8] Altera Nios Tutorial, <http://www.altera.com/literature/lit-nio2.jsp>, visited: 12/31/2005
- [9] Altera DE2 Development Kit, <http://www.altera.com/education/univ/materials/boards/unv-de2-board.html>, visited: 1/6/2006