Design and Implementation of a Minuscule General Purpose Processor in an Undergraduate Computer Architecture Course

Archana Chidanandan J.P. Mellor Laurence D. Merkle
Rose-Hulman Institute of Technology
Terre Haute, Indiana 47803

Abstract

This paper describes a project for an introductory computer architecture class. The project involves designing and implementing a "miniscule" processor that can run programs within certain boundaries. In this paper, we describe the goals of the project, the process, and the challenges faced by students in successfully completing the project.

1. Introduction

At our school, computer architecture concepts are taught over two 10-week terms, with 40 contact hours per term. The first course covers assembly language programming and basic computer architecture principles. The second course covers advanced topics such as pipelining, caching, scheduling, etc. In this paper, we will describe the term-long project that the students in the first course must complete in order to pass the class.

2. Project description and goals

The project requires the students to design a "minuscule" instruction set general purpose processor that can execute programs stored in an external memory. Students also model their design using electronic design automation (EDA) software, test it through simulation, assess its performance, synthesize it, and implement it on a Field Programmable Gate Array (FPGA) microchip. The goals of the project are that students should be able to:

a. Apply the principle of abstraction in analysis and design problems.

b. Specify, design, test, and document instruction set architectures and their hardware implementations, taking into consideration key computer organization design principles.

c. Work effectively as members of a team.

The project is typically assigned over a 10-week period and passing the project is a requirement for passing the course. Students work in teams of three to four.

Students are provided with a sample high-level language program that their architecture must be able to run. Students must convert the program to their assembly language and subsequently their machine language. The program requires some arithmetic and logical operations, conditional statements, loops, and implementing and invoking a recursive procedure. In addition, the inputs are obtained through interrupt-driven I/O and students must implement some simple interrupt-service routines. Students are encouraged to write at least one other program to test their architecture.

3 Integration of the project into the course

In order to enforce the use of abstraction in the design process, similar to the approach in the text[1] students begin by designing an instruction set architecture(ISA), followed by the design of the hardware and its organization to implement the ISA.

Milestone 1: In the first milestone, the team is required to design their own unique ISA, i.e. the assembly language and machine language specification. A requirement is that the instructions cannot be all 32-bit in length. At this point, the students have been exposed to the MIPS architecture's assembly and machine language and basic ISA design principles. Teams are encouraged to explore variable-length instructions, stack-architectures etc.

Milestone 2: In the milestone, teams must describe an implementation of their ISA using Register Transfer Language (RTL). In class, students would have seen the RTL specifications for the single-cycle and multi-cycle implementation of a subset of the MIPS ISA. Teams are typically encouraged to design a multi-cycle implementation as this will allow them to explore the complexity in design of the control unit.

Milestone 3: At this stage, teams must design the datapath for their architecture. They must complete the component list and correctly identify all the control signals that

2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)
0-7695-2849-X/07 $20.00 © 2007 IEEE

are required to implement the datapath. Teams are also encouraged to start implementing and testing the components. They must also devise a plan to integrate the components to eventually implement the datapath.

Milestone 4: Now that all the control signals have been identified, the teams must design the control unit. They can either use a FSM or a micro-programmed control unit. Again, students would have been introduced to control unit design principles in class.

Milestone 5: In this final stage, students must complete the implementation of the datapath, include the control unit and also integrate the provided memory into the implementation. They must test the implementation and when possible, students are encouraged to implement the design on an FPGA.

Deliverables: For each milestone, students write a design report that summarizes their design choices and provides the details of their design. They also maintain and submit a design journal that explains their design choices at each stage. Students will also eventually turn in the implementation files. They will write a summary report that describes their design process, lessons learned, and the performance characteristics of their architecture. In lieu of a final exam, students present their architecture to their peers and evaluate each other's designs.

4. Tools Used

In the initial offerings of the course, LogicWorks[2] was used to implement the project using the schematic entry tool. Since then, Xilinx ISE foundatiion[3], ModelSim, Cadence NC-Sim[4], and SimView have been introduced. The use of HDL such as Verilog is also encouraged.

5. Challenges

The course is a required for Computer Science, Computer Engineering, and Software Engineering students. The pre-requisites for the course are a course in logic design and basic programming concepts. While these requirements are sufficient for students to understand the concepts, they do not always meet the requirements to complete the project implementation. Until last year, students had not been exposed to most of the CAD tools. This required that about 10 hours of class time be devoted to getting students comfortable with the tools and HDL such as Verilog. In the past year, the pre-requisite logic design course has switched to using Cadence and SimView.

Another challenge is the sheer scale of the implementation. Students have had little experience with the implementation of such a large project. They did not have sufficient experience to understand the need for incremental implementation and testing. This has often resulted in a variety of problems during the implementation which the students are unable to solve soon enough and hence resulting in incomplete implementations.

A third challenge lies in the ISA design stage. Since students would have been exposed mainly to the MIPS architecture, it was difficult to wean them from it and have them come up with unique architectures.

The last two items listed above have been addressed to some extent as a result of the changes in the logic design course. Because, we no longer have to spend as many hours on teaching the students the tools, we are able to use that time to teach students about at least one other architectural style in some detail. In the current offering of the course, students initially see a stack-based architecture. They are also given a partial implementation of the architecture and are then required to complete the implementation. The "mini-project" gives students some experience with working with a large-scale implementation before they implement their own architecture from scratch.

6. Conclusion

The project gives the students a hands-on experience with processor design. Students typically enjoy the project and have commented on how much they learn. We have typically achieved around a 50% success rate in the implementation phase. We do not view this as necessarily a failure in the project itself, as implementation is just one phase of the project. We are consistently surprised by the ISAs that students design and by the risks they are willing to take in their design which sometimes work out well and sometimes don't. However, in either case, they learn some valuable lessons. We hope with the new changes we have described above we will be able to increase the success of our students at the implementation stage also.

7. Acknowledgements

The authors would like to acknowledge the generous donations of EDA tools and FPGA boards from Xilinx Corporation and Digilent Inc. and the contributions of Dr. Rimli Sengupta, who pioneered this project at Rose-Hulman.

References

[1] David.A. Patterson and John L. Hennsessy, "Computer Organization and Design - The Hardware/Software Interface," 3rd ed., Morgan Kaufmann.

[2] http://www.logicworks4.com/

[3] http://www.xilinx.com/

[4] http://www.cadence.com/