

Teaching Computer Organization with HDLs: An Incremental Approach

John A. Nestor

*Department of Electrical and Computer Engineering
Lafayette College, Easton, Pennsylvania 18042
nestorj@lafayette.edu*

Abstract

This paper describes the use of Verilog HDL in a series of design projects for an undergraduate Computer Organization course. Students are given Verilog “working models” of pedagogical designs that can first be simulated to enhance initial learning and then extended and modified to develop more in-depth understanding. Projects include adder/ALU design and processor design using the single cycle, multicycle, and pipelined processor implementations presented in the popular Patterson & Hennessy text [1]. This incremental approach allows students to focus on the underlying concepts of the course as they become more familiar with Verilog. The models and supporting project assignments are available online at <http://foghorn.cadlab.lafayette.edu/ece313/>.

1. Introduction

The Verilog [2] and VHDL [3] Hardware Description Languages (HDLs) have replaced schematic diagrams as the foundation for digital systems design. As a result, HDL-based design has become an essential topic in Electrical and Computer Engineering Curricula. In upper-level courses, HDL modeling and simulation provide opportunities to enhance student understanding of complex topics.

In Computer Organization and Architecture courses, HDLs are typically used in two ways:

- *High-level behavioral modeling.* Describes the function of different architectures, but not implementation (e.g., [4]).
- *Register Transfer Level (RTL) modeling.* Describes the structure and function of processors at the implementation level.

Behavioral modeling is often used in Computer Architecture courses to explore high-level features, while RTL modeling is often used in Computer Organization courses to describe specific

implementations. Most Computer Organization courses use RTL modeling in a “design from scratch” approach in which students design an entire processor from a subset of a common Instruction Set Architecture (e.g., [5], [6]). While this approach gives students a strong understanding of processor design at the end of the project, it can be difficult for them to gain an initial understanding as the concepts are first presented.

This paper describes a different approach based on the idea of “working models”. Instead of expecting students to immediately start using HDLs to create designs from scratch, HDL descriptions of complete pedagogical designs are distributed to students. Students first simulate these models to gain an initial understanding of the Computer Organization and the use of HDLs for RTL modeling. Later, they extend and modify these designs to enhance their understanding while developing debugging skills.

These “working models” are based on pedagogical designs presented in Patterson & Hennessy’s text *Computer Organization and Design* [1] (COD) and include a ripple adder and *single-cycle, multicycle, and pipelined* processor implementations. The recently released third edition of COD includes on CDROM a Verilog tutorial, behavioral models of the multicycle and pipelined designs, and an RTL model of the multicycle design. The models described here complement this new material by providing RTL models of all three processor implementations in a consistent style that matches the structure of the designs as described in the text.

2. Verilog Concepts

The Computer Organization course in the Electrical and Computer Engineering program at Lafayette College is normally taken by first semester Juniors. It has two prerequisite courses which cover digital design fundamentals including some basic Verilog.

The Computer Organization course builds on this foundation by providing an overview of Verilog RTL coding that emphasizes design for synthesis. This overview requires the equivalent of three 50-minute lectures and includes the following topics:

- Basic syntax; modules and ports.
- Wires, data types, and operators.
- Hierarchy and module instantiation.
- Combinational logic using simple expressions (`assign`) and procedural code (`@always`).
- Sequential logic using procedural code (`@always(posedge clk)`).
- Finite state machines.

3. RTL Models and Projects

Design examples are distributed to students as RTL models attached to project assignments. Each RTL model is coded to clearly and concisely present the structure of an example described in the text. Building blocks are described as separate modules that are instantiated in a top-level module, and common building blocks are shared between designs. The following summarizes each model and associated project assignment. They are also available online at <http://foghorn.cadlab.lafayette.edu/ece313>.

Project 1 – Adder/ALU – the initial model (26 lines of Verilog) describes an 8-bit ripple adder. The assignment requires that students extend this initial design to 16 bits, implement group-carry lookahead, create an ALU design corresponding to that discussed in Section B.5 of COD, and finally modify the ALU design to use group carry-lookahead.

Project 2 – Single Cycle Processor – the initial model (518 lines of Verilog) describes the single-cycle implementation discussed in Section 5.4 of COD. The project assignment requires that students extend this design to implement additional instructions.

Project 3 – Multicycle Processor – the initial model (658 lines of Verilog) describes the multicycle implementation discussed in Section 5.5 of COD. The project assignment requires that students extend this design to implement additional instructions and an undefined-instruction exception.

Project 4 – Pipelined Processor – the initial model (697 lines of Verilog) describes the pipelined implementation discussed in Sections 6.2-6.3 of COD. The project assignment requires that students extend this design to implement additional instructions and either a) data forwarding with stalling

on load-use hazards; or b) branch in the instruction decode stage with pipeline flushing.

The small size of the models allows them to be used with evaluation or student versions of commercial simulators. This allows students to work on these projects using their own computers if desired.

4. Conclusions

This paper has described the use of Verilog to reinforce the concepts of a Computer Organization using an incremental approach in which students extend and modify existing models. Student response to this effort has been enthusiastic; they indicate in comments that the projects have greatly enhanced their understanding of the material. The Verilog skills which students learn in this course are used in later courses which involve FPGA and VLSI design.

There are several directions in which this work could be extended. First, the scope of the design projects could be increased by requiring the implementation of more difficult features. Second, the models could be used as the basis for FPGA-based prototyping as in [7], [8]. Finally, additional models could be introduced to help students understand additional topics in Computer Organization, including caches, virtual memory, and input/output.

References

- [1] D. Patterson and D. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 3rd Ed., Morgan-Kaufmann, 2004.
- [2] D. Thomas and P. Moorby, *The Verilog Hardware Description Language*, 5th Ed., Kluwer, 2002.
- [3] S. Yalamanchili, *Introductory VHDL: From Simulation to Synthesis*, Prentice-Hall, 2001.
- [4] D. Hyde, "Teaching Design in a Computer Architecture Course", *IEEE Micro*, Vol. 20, No. 3, May/June 2000, pp. 23-28.
- [5] G. Manimaran, "CprE 305 Computer Organization and Design" Homepage, Iowa State University, <http://vulcan.ece.iastate.edu/~gmani/cpre305/>.
- [6] S. Hauck, "EE 471 Computer Organization", University of Washington, <http://www.ee.washington.edu/class/471/hauck/>.
- [7] J. Hamblen, "Rapid Prototyping Using Field-Programmable Logic Devices", *IEEE Micro*, Vol. 20, No. 3, May/June 2000, pp. 29-37.
- [8] M. Holland, J. Harris, and S. Hauck, "Harnessing FPGAs for Computer Architecture Education", *International Conf. on Microelectronic Systems Education*, June 2003.