

Using Verilog HDL to Teach Computer Architecture Concepts

Dr. Daniel C. Hyde

Computer Science Department
Bucknell University
Lewisburg, PA 17837, USA

hyde@bucknell.edu

June 27, 1998
Workshop on Computer Architecture Education
Barcelona, Spain

Introduction

- Best way to learn how a computer works is to **design** one.
- Impractical to design physical hardware for sophisticated computer architecture concepts, e. g., cache.
- Undergrads, especially, need to be active learners.
- Use Verilog (a hardware description language) to describe, design, run and test.
- We describe web-based course materials for using Verilog to teach computer architecture concepts.

Institutional Context

Bucknell University

- High quality liberal arts college with attached engineering college. Modest MS program.
- Central Pennsylvania, U. S. A.
- 3300 Undergraduates; 25/year in CS

Computer Science at Bucknell

- Three CS degrees BA, BS and BSE
- Two-hour structured lab for almost every CS course
- Faculty staff the labs

Computer Architecture Course (CSCI 320)

- Prerequisite - computer organization course
- Taken by mostly CS seniors
- 13 two-hour structured laboratories.
- Verilog simulator on Sparc workstations
- Text - *Computer Architecture: A Quantitative Approach*, 2nd ed., by Hennessy and Patterson.
- Focus on design
- Large design component satisfies US Accreditation Board for Engineering and Technology (ABET)

Importance of Laboratories for Learning

At Bucknell, labs are an important component of the students' learning environment.

With Laboratory exercises, students

- solve “constrained” designs - still many paths to solution.
- learn differently than from reading, listening to lectures or doing homework problems.
- can reinforce concepts learned in reading and lectures.
- can explore details and key issues.
- are allowed to place their “hands on” the tools and techniques which provides more intuitive feel for the material and a sense of ownership of the material.
- can build a system that they can expand or analyze.
- can play “what if” scenarios.
- can explore beyond the assignment.

Hardware Design Languages (HDL)

Long Tradition of Use in Industry

- Babbage (1840)
- von Neumann - ISA (1940s)
- Iverson's APL - originally to describe IBM 360 (1960s)
- many others

Today Two Major HDLs in Industry

- VHDL
- Verilog HDL

Used by Education

- AHPL - Hill and Peterson (1970s)
- ISP - Bell and Newell (1971)
- RTN - Heuring and Jordan (1997)
- many others for description of digital systems

Some HDLs do more than describe a digital system.

- Simulators to analyze and test designs.
- Verilog HDL - **FREE** simulators available.

VHDL Verses Verilog HDL

VHDL (“V” short for VHSIC)

[Very High Speed Integrated Circuits]

- Designed for and sponsored by US Department of Defense.
- Designed by committee (1981- 1985).
- Syntax based on Ada programming language.

Verilog HDL

- Designed by industry.
- Designed by individual (Philip Moorby 1985).
- Syntax based on C programming language.
- Dominant HDL; hardware designers like it.

Both IEEE Standards.

For Educator - easy choice between the two!

Why Verilog? Why not C or C++?

Verilog has

- Several levels of abstraction - allows expression of behavior deferring the details to later stages of design.
 1. behavior level (register transfers)
 $AC \leq AC + MD;$
 2. structural level, e. g., carry ripple adder
 3. gate level
 4. switch level (pass transistors)
- Concise/precise descriptive power of digital systems.
- Good control of output.
- Continuous assignment - used to model combinational logic.
- Structured language like Pascal and C++
if-then-else, case, while, for, repeat constructs.
NO gotos which allow spaghetti code.
- Good timing facilities - important for performance measurement
- Features for concurrency
 - several register transfers at same time
 - multi-threading within a system (one control unit)
 - multiple systems (more than one control unit)

Verilog has great flexibility for the educator - allows student to grow in language knowledge as semester progresses.

Example Verilog Code (Behavioral Level)

```
reg [0:31] AC, MD, IR;
reg [0:9]  MA, PC;
reg [0:31] MEM [0:1023];

// instruction fetch
#1 MA <= PC;
#1 MD <= MEM[MA]; // memory read
#1 IR <= MD; PC <= PC + 1;

// decode and execute code for a LOAD
#1 if (IR[0:3] == 4'b0000) begin
    #1 MA <= IR[22:31]; // last 10 bits address
    #1 MD <= MEM[MA];  // memory read
    #1 AC <= MD;
end
```

- Assume modeling with trailing edge D-flip flops.
- The #1 means delay one clock period.
- The <= means **blocking** assignment.
- Natural representation of registers and register transfers.

How Verilog is Used in Computer Architecture Course

- Goal –to formulate a computation model for digital systems using *small* subset of Verilog.
- Model is much higher level of abstraction than digital circuits.
 - Students no longer need to think in digital circuits.
 - Appreciated by non-hardware type CS majors.
- Students learn how to “compile” Verilog-based model to both **data unit** and **control unit**. Shown that it can be automated.
- Students are *told* that with automated tools, the Verilog-based model could be translated to ICs.
- Verilog-based model used in lectures.
- In labs, students design with Verilog-based model and test with simulator.
- Some lab exercises of the 12 available
 - simple four instruction accumulator CPU
 - redesign accumulator CPU to stack and register machines
 - instruction lookahead - 2 stage pipeline
 - design at structural level an 8-bit adder from gates
 - add cache memory and analyze timing
 - multiple systems and signaling between them
 - floating point adder

Conclusions

- Students find approach easy to relate to Hennessy and Patterson's textbook. Good complement to text.
- Using major industrial HDL highly motivating to students.
 - hardware-types see learning Verilog important for career opportunities.
 - software-types see Verilog as another programming language to learn.
- Verilog-based model and approach allows students to design architectural components and test them. Enhancing learning.
- Cheap to use - free simulator and free web-based materials
 - 32 page Verilog manual
 - 12 page paper on Verilog-based computational model.
 - 12 laboratory exercises (solutions available)
- Over dozen universities currently using materials.
- This approach allows computer architecture instructors the ability to teach computer architectural concepts.
 - with flexibility and freedom to modify the approach to integrate into current instructional needs.
- Web Site for materials

<http://www.eg.bucknell.edu/~cs320>