LEVERAGING HARDWARE DESCRIPTION LANGUAGES AND SPIRAL LEARNING IN AN INTRODUCTORY COMPUTER ARCHITECTURE COURSE

John H. Robinson and Ganesh R. Baliga
Computer Science Department
Rowan University, Glassboro, NJ
856 256-4805
Email: robinsonj@rowan.edu, baliga@rowan.edu

ABSTRACT

This paper describes our experience with using hardware description languages (HDLs such as VHDL) in the sophomore level Computer Architecture course for Computer Science (CS) majors. In our approach, we leverage the students’ understanding of Object Oriented Programming (OOP) to introduce programmatic specifications of computer components such as multiplexors, adders etc. Students will be exposed to HDLs yet again in more advanced hardware courses such as Digital Design, Computer Architecture and Embedded Systems Programming. In this manner, we plan to harvest the benefits of spiral learning, whose effectiveness in pedagogy is well accepted. The proliferation of Fully Programmable Gate Arrays (FPGAs) and modern development environments makes it feasible for CS majors to experience the creation of functioning hardware using specifications coded in VHDL. We expect our approach will result in improved learning outcomes for CS majors in the hardware field.

INTRODUCTION

As part of a general revitalization of the curricula associated with undergraduate computer architecture, we proposed a project to integrate modern digital hardware design technology and build stronger integration among hardware courses for CS majors that have been traditionally taught in isolation [2]. The ties between the courses will be based on spiral learning techniques of redundancy and increasing complexity [6]. A hardware description language, in particular Very High Speed Integrated Circuits Hardware Description Language (VHDL), was used to introduce recurring topics iteratively in a constructive way. Iterative reintroduction using VHDL over the entire hardware curriculum will allow the students to leverage the knowledge gained as a CS major and apply it to understanding computer architecture and the software that drives the hardware. We expect that such an iterative approach reinforces understanding since it allows the students to build upon their prior knowledge and, furthermore, to approach the very same concepts from different perspectives and increasing complexity. This iterative method is inspired by the idea of spiral learning [6] which builds on a constructivist view on learning. It assumes that learning is an active process in which learners construct new ideas or concepts based upon their prior knowledge. A curriculum can support this process if it revisits its basic ideas repeatedly.
Traditionally, a computer hardware curriculum teaches theoretical knowledge and employs simulation software to run projects. The revitalization project not only focused on introducing and integrating new digital design technology into the curricula, but also, moving the curricula from a theoretical/simulation based focus to a project based learning environment and providing continuity over the entire hardware course sequence. The impetus for this initiative is to improve, expand, and facilitate student learning about digital design, computer circuits and hardware, while capitalizing on the students’ existing strengths in Object Oriented Programming (OOP) techniques. Our approach will include using the VHSIC Hardware Description Language (VHDL) language, Field Programmable Gate Arrays (FPGAs), and CAD software from Aldec and Xilinx.

Current trends in industry [11] demand that CS graduates have a fluency in modern digital systems design techniques and become agile product development and rapid prototyping team members. There is also a convergence of hardware design methodologies (Software Based) currently used by Computer Scientists and Electrical Engineers. Modern digital design techniques (Software Based) are essential to both CS and Electrical Engineering curricula [14,15,16]. This provides a unique opportunity for CS educators to produce graduates that are comfortable working in a realm traditionally reserved for Electrical Engineers and demands that Electrical Engineering curricula become more software centric [1,3,7,8,9,11,13]. This novel approach to digital systems design will allow the students to apply the knowledge gained as a CS major to the design and understanding of computer architecture.

Leveraging what CS students learn and employing a spiral curriculum will facilitate the common threads pulled though the entire CS hardware curriculum. This paper describes the experience of using VHDL in an introductory computer architecture course.

THE NEW CURRICULUM FOR COMPUTER ORGANIZATION

Traditional CS OOP methodologies, project driven learning, and visualization techniques will be utilized as a novel approach to allow the “natural” exploration of the details of implementation for the “black box” components introduced in computer organization. Projects and lectures in this course allow the students to apply their CS knowledge in understanding computer hardware components and the software that allows communication between the components. An introduction to VHDL and FPGAs will allow the students to examine the details of implementation and experiment with the “black box” components taught in Computer Organization. A specific mini MIPS processor will be used in this course and will be presented over the entire hardware course sequence. Computer Organization (our sophomore level Computer Architecture course) will introduce the architectural features of this processor.

The novel curriculum for teaching computer architecture is presented to the students using VHDL and concept sets. These concept sets are tools for learning about digital design and modern design techniques. The concept sets aim to help all CS undergraduate students achieve the following targeted learning outcomes:

- Move from a theoretical to a project based learning environment using state-of-the-art techniques for digital design, e.g. HDLs, FPGAs, and computer simulation.
- Design hardware using traditional CS design techniques and modeling.
- Better integration of courses over the entire computer hardware curricula.
- Deeper appreciation of the boundaries between hardware and software.
- Student skills and knowledge that is transferable to subsequent laboratory courses and future careers.

USING VHDL TO TEACH COMPUTER ORGANIZATION

In the Fall semester of 2009, a new approach to teaching Computer Organization was piloted. The course moved from a purely theoretical/simulation format to a combination of theoretical/simulation and project based learning utilizing the concept set and VHDL. The concept set is a series of projects that explore the hardware of a microprocessor using VHDL and OOP. New software was also introduced to provide access to modern simulation and development tools.

Traditionally, Computer Organization is taught using microprocessor models and the accompanying instruction sets, simulation software, and assembly programming. The course serves as an introductory experience to the concepts of computer architecture. In this version of the course, a MIPS processor was introduced along with VHDL as a way to illustrate the details of the components that comprised the processor as outlined by our concept sets. The MIPS processor was deconstructed and each component (multiplexers, registers, ALUs, etc.) was described in VHDL and discussed. The processor was reconstructed as a controller and datapath. The instruction set for the processor was also explored and simulated using PCSpim [12] and PathSim [10]. The similarities between OOP techniques, high level languages and VHDL were emphasized and simulations of the VHDL code running on physical hardware were shown using software from Aldec and Xilinx. The final concept set project had the students implement a Fibonacci number generator machine in VHDL using the algorithm, state diagram, and the datapath shown in Figures 1, 2 and 4 [17]. The expected VHDL code, showing the code for the loop driving the datapath, is shown in Figure 3. A balance between theory/simulation and hardware design was sought using the concept sets and VHDL. The effort to revitalize the hardware curriculum seeks to utilize this balance over the entire course sequence.

```plaintext
if (input == 0)
    output = 0;
else if (input == 1)
    output = 1;
else {
    F1 = 0;
    F2 = 1;
    while (input >= 2) {
        output = F1 + F2;
        F1 = F2;
        F2 = output;
        input = input – 1;}
}
```

Figure 1 – Fibonacci Algorithm
architecture behv of FSM is
-- define enumerated states
type states is(idle, init, doLoop, done);
-- declare state signal
signal nState, cState : states;

process(start, comp_result, cState)
begin
case cState is
when idle =>
    -- code to drive components in datapath
when init =>
    -- code to drive components in datapath
when doLoop =>
    N_Sel <= '0';
    N_ld <= '1';
    F1_Sel <= '1';
    F2_Sel <= '1';
    F1_ld <= '1';
    F2_ld <= '1';
    Out_enb <= '0';
    if comp_result = '1' then  -- done
        N_Sel <= '0';
        N_ld <= '0';
        F1_Sel <= '0';
        F2_Sel <= '0';
        F1_ld <= '0';
        F2_ld <= '0';
        nState <= done;
    else
        nState <= doLoop;   -- Loop
    end if;
when done =>
    -- code to drive components in datapath
end case;
In this project students were introduced to the complexities of implementing a controller for a small application specific processor. Concepts such as state diagrams, enumerated types, decisions, and switch statements allowed the students to implement the controller by leveraging knowledge acquired in other CS courses.

The use of VHDL allowed the exploration of the components and subsystems. The use of VHDL is “natural” to CS majors because it mimics OOP techniques where the details of implementation are typically ignored and complexity is managed with the use of classes and objects. The complexities of the individual components are hidden or “encapsulated” in the VHDL code allowing larger systems to be built by drawing simple schematics to connect these components. The topics presented in this version of the course were the first level of the spiral curriculum [6] model for the hardware courses. The concepts set will be revisited in each of the subsequent hardware courses (Principles of Digital Computers, Digital Design Lab, Embedded Systems, and Advanced Computer Architecture) of the CS major. Each exploration of the concepts will be at a higher level of complexity than prior courses, reinforcing what was introduced in latter.

**SUMMARY AND CONCLUSIONS**

These days CS majors are taught software design using techniques featuring Object Oriented Programming, code reuse, rapid prototyping, project centered learning, and visualization. In addition, they are taught the theory, components, and design of computer hardware. Traditionally electrical engineers did hardware design, but with the advent of FPGA and HDLs, computer scientists can also design hardware. There is a convergence of hardware design methodologies used by electrical engineers and computer scientists. This provides a
unique opportunity for CS educators to produce graduates that are comfortable working in a realm traditionally reserved for electrical engineers. A learning experience featuring spiral learning [6] and a methodology that is “natural” to CS majors will allow them to develop a deeper understanding of digital systems. This novel approach to digital systems design will allow the students to apply the knowledge gained as a CS major to the design and understanding of computer architecture. The curriculum described in this paper will enable students to bridge the gap between hardware and software design to produce a unified view that enables the co-development of systems using both hardware and software.

REFERENCES