INFRASTRUCTURE AND COURSE PROGRESSION FOR COMPLEX IC DESIGN EDUCATION

Paul D. Franzon, Wentai Lui, Clay Gloster,
Toby Schaffer, Alan Glaser, Andy Stanaski

Department of Electrical and Computer Engineering
Box 7911
North Carolina State University
Raleigh, NC 27695-7911
paulf@ncsu.edu

ABSTRACT

The ability to cope with design complexity is an important skill for computer engineers, especially potential System On a Chip design engineers. Complexity has many facets, including gate count, the ability to handle multiple disciplines simultaneously, and the ability to cope with complex CAD tools. Teaching complexity also requires considerable investment in tool flows, design examples and tutorials. Here, the approach used at North Carolina State University will be described and illustrated.

1. INTRODUCTION

A core skill required of competent computer engineers, especially those designing tomorrow's Systems on a Chip, is the ability to cope with complexity. Teaching a student to cope with complexity involves addressing multiple issues including:

- Design size, in terms of gate count, module count, etc.
- Design complexity, in terms of the disciplines needed, including hardware, software, and applications knowledge.
- Verification complexity, i.e., learning how to verify correctness and debug complex and subtle designs.
- Dealing with CAD tools. Modern CAD tools lend tremendous power to designers but are often more complex and buggy than the design itself. For example, our installation of tools from Cadence Design Systems requires 4 GB of storage space and our list of "known problems and workarounds" gets longer every year.

In this paper, we describe our course progression, which is intended to teach students how to cope with complexity, and part of our supporting infrastructure.

2. UNDERGRADUATE COURSE STRUCTURE

We start specifically dealing with complexity in a required junior-level course, ECE 342 – Design of Complex Digital Systems. The capstone project in this course requires students to design a small MIPS microprocessor using a mixture of Verilog HDL and schematic capture within the Cadence design environment. Our experience is that using commercial tools doubles the effort required for the laboratories (for both students and instructors) but pays off enormously for the students in terms of an improved ability to cope with complexity and a marketable skill to mention on their resume.

The infrastructure required for the laboratory in this course took about 6 TA-months to develop (not counting actual contact and grading time) and takes another 2 TA-weeks per year to maintain.

This course feeds into a number of senior electives, including:

- ECE 460 – Digital Systems Interfacing
- ECE 463 – Computer Design and Technology
- ECE 491B – Embedded Systems
- ECE 491T – Optimizing Compilers

3. GRADUATE EDUCATION

Our graduate education expands the use of real-world CAD tools to include several IC design tools from a variety of vendors, in particular Cadence and also Synopsys (Design Compiler/Design Analyzer), Avanti (HSPICE), and Ansoft (Maxwell 3D). Our philosophy is not to turn our courses into training courses for the tools but to use the tools so students learn how design work is done in a modern CAD tool environment. Our argument is that this is necessary because the constraints imposed by the CAD tools can influence the design and verification approach even more than basic design skills do.

The core courses in this sequence are:

- ECE 520 – ASIC Design
- ECE 746 – VLSI Design

In both courses, tools from Cadence, Synopsys and Avanti are used to illustrate and implement the design steps as well as convey the importance of "CAD-tool-aware" design management and design flows.

The students are then encouraged to pursue courses in each of three areas:

1. Applications (DSP, communications, networking, etc.)
2. Systems (computer architecture, RTOS, programming)
3. Advanced Design (FPGAs, DFT, etc.)

The CAD-related materials used in these courses required about three man-years of investment. Fortunately, we needed to get these tools working for our research projects, so it was those projects that provided the effort — a perfect example of the marriage of teaching and research!

Besides a number of tutorials and scripts, our infrastructure consists largely of a design kit which was put together to support the scalable MOSIS ruleset for IC design within Cadence. This kit and associated flow tools took most of the effort and is described next.

4. CDK ORGANIZATION

Our Cadence environment has been customized with several technology files and a fair amount of custom SKILL code. These files contain information useful for full-custom CMOS IC design via MOSIS. Some of this information includes layer definitions (e.g. colors, patterns, etc.), parasitic capacitances, layout pcells, SPICE simulation parameters, Diva rules for DRC, extraction, and LVS verification, and various GUI enhancements. This environment is called the NCSU Cadence Design Kit (CDK) and is freely available via the Web for download and use by any institution (see the URL at the end of this paper). Below is a very brief description of the CDK contents.

- Technology Files and Diva Rules Files

  The technology files included in the CDK define the mask layers and their appearances and properties, as well as parameters used at library creation time which set the value of lambda, the technology code, and the availability of process-dependent layers (layers that are not common to all MOSIS processes).

  The structure of these files follows the flow described in the OpenBook section *Technology File and Display Resource File User Guide*.

  Verification (DRC, circuit extraction, LVS) is done with Diva, which is controlled by included rules files.

- Technology Libraries

  One technology library is provided for each MOSIS CMOS process which supports the SCMOS rules. Usually, design libraries are linked to the appropriate technology library to obtain process-dependent parameters.

- Standard Parts Libraries

  The CDK includes libraries for standard parts. These libraries contain common analog and digital parts symbols (e.g., transistors and RLC components), Verilog primitives (e.g., logic gates), and example sheet borders.

- Device Models

  The CDK includes transistor models (HSPICE level 13/Spectre level 4 and HSPICE level 49/Spectre level 11) for most MOSIS SCMOS processes, which are obtainable from the MOSIS web site.

- SKILL Code

  SKILL is the Lisp-like language that is used to interface with the Cadence design environment, from constructing new GUI elements to modifying the design database directly. The CDK includes a variety of custom SKILL code — forms, menus, CDF callbacks, and pcell definitions. This code, along with the Diva rules, provides the large majority of the CDK's added value.

4.1. CDK Functionality

The CDK provides customizations for the Design Framework, Composer, Analog Artist, Verilog, Virtuoso and Diva. See our Web page (URL given below) for more information.

5. CONCLUSIONS

This paper presents an overview of our "CAD-tool"-centric approach to teaching IC design and some of the supporting infrastructure. You can find out more about and download the CDK from `www.ece.ncsu.edu/cadence/CDK.html`, and there is class information at `www.ece.ncsu.edu/lockers`.