Active Learning Techniques in a CAD Course

Scott Hauck <hauck@ee.washington.edu>
Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA

Abstract

Physical Design is a complex CAD topic, which can be challenging to teach to electrical and computer engineers. This paper covers simple techniques for adding active learning into a CAD classroom. These techniques are easily integrated into offerings at other institutions.

Introduction

I teach a course on Physical Design called EE541: Automated Layout of Integrated Circuits. This course, intended primarily for graduate students in the electrical engineering (and some computer science), serves as a quick introduction to most aspects of physical design. The goal is to quickly get students conversant on the major classical algorithms in the field, and how the individual pieces fit together into an integrated whole. The course covers Partitioning, Floor-planning, Placement, Global and Detailed Routing, Compaction, and Retiming. It runs for 2 hours a day, twice a week, for 10 weeks.

The tendency in a course like this is to show the techniques in their fully-formed versions, meaning students don't grasp the process of how to create these algorithm, or how to approach new problems. A solution I have adopted is that of active learning; simply put, find ways for students to teach themselves the concepts, instead of just receiving the information as given from the lecturer. Note that while active learning techniques do form the starting point of all topics, via somewhat ill-constrained optimization problems done by students in class, and the end-point, via programming assignments, lecture is still used for perhaps 90% of the class time.

The active learning techniques split into three major categories, each of which will be described in the sections that follow: programming assignments in Java via the Aphyds framework, pencil-and-paper optimization problems done in class, and wooden-block based optimization problems also done in class.

CAD Programming: Aphyds

Just as many others do, I ask my students to program up several classical algorithms. This is done within the Aphyds framework [Hauck03], a 14k line Java program that includes file I/O, graphics, basic data structures, and the wrapper for the algorithms. Students put in the guts of the major algorithms, each of which involves 10-20 hours of programming and about 100-200 lines of code. The programs are: (1) static timing estimator (used as an easy introduction to Java and Aphyds for the students); (2) Fidducia-Mattheyses bipartitioning; (3) Floorplanning sliceable floorplans with variable node sizes; (4) Simulated Annealing placement; (5) A maze global router; (6) channel routing via the left-edge algorithm.

The completed Aphyds system is also given to students in an encrypted, compiled Java JAR. This allows students to check their work, and also serves as demonstrations during lecture to help show students how the algorithms work. Details of Aphyds are presented in [Hauck03]. The code is freely available to other educators.

Pencil-and-Paper Exercises

Although complete CAD algorithms are complex optimization tools, most of the basic concepts are quite intuitive for students, and students' natural approaches to these problems are generally very good. Often all that is necessary is some time to reason through the problem to get a solution. However, traditional lectures do not generally give students this time.

To deal with this, before starting any of the major sections of the class I first give the students a simple, often ill-defined, problem to solve in class. The simplest are paper-and-pencil optimization problems, shown in figures 1-3. Specifically, I hand out to small groups of students a single sheet of paper with an optimization problem on it, without explaining to students how these problems are "really" solved. They are given about 20 minutes to solve the problem in teams. During that time I circulate through the classroom answering questions, though typically the answer is "what do you think is a reasonable way to approach that problem", or "if this was an electronic circuit, what would you do"; this is generally sufficient. I make notes on approaches I've seen students taking, and problems they have encountered. These notes become a 5-minute wrap-up session, in lecture format, on what the students have done. This wrap-up is an excellent introduction to the topic areas, since most of what students stumbled upon is the very subject matter I planned to cover. Students are excellent at recreating most of the major discoveries in CAD algorithm if given time to do so.

Consider my problem for floorplanning. Students are given a piece of graph paper, and the instructions:

> "*Please create a layout of an apartment with four rooms: Kitchen 4x5, Bathroom 3x5, Living/Dining Room >=24 squares, Bedroom >=30 squares. All rooms are rectangular. The apartment should have the smallest possible rectangular area.*"

Students quickly realize that there are communication needs in the layout – the dining room and kitchen should generally touch, and access to the bathroom from the dining room and kitchen is important. They then generally

do multiple solutions, evaluating different approaches, especially in the face of the variable room size constraints.

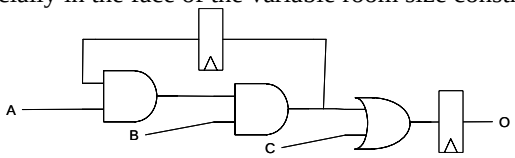

Figure 1. Retiming exercise. "Make the following circuit have as high a clock rate as possible. All gates have a delay of 1ns, all registers have 0 delay".

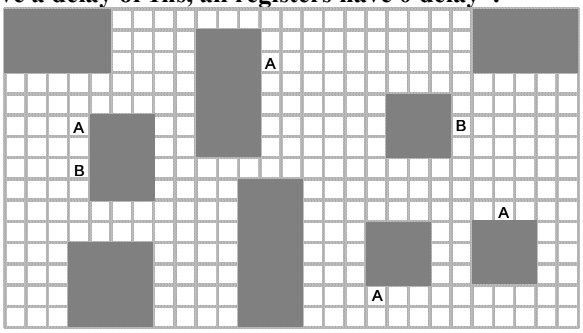

Figure 2. Global Routing. "Route together all the A's, and all the B's, minimizing the amount of metal. Dark areas are impassible barriers".

For the retiming problem, students can consider not only register movement (push the register on O back one gate), but also logic restructuring (remove an A*B from the feedback path) and repipelining (add another register to all inputs). These are natural discoveries that students easily make.

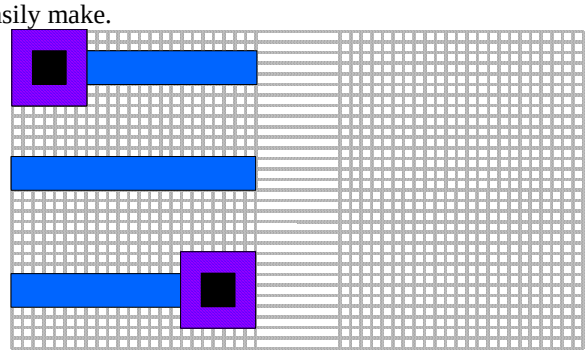

Figure 3. Compaction. "Minimize the channel height. Minimum 3λ spacing".

Wooden Block Based Exercises

Some problems, such as placement and partitioning, are best solved by an iterative approach, which is difficult to simulate via paper and pencil. Solving each problem generally involves starting with an initial, poor solution, and then making small improvements until an overall good result is achieved. Students naturally adopt this approach themselves. To support this, I have created a physical analog of a netlist out of high-tech components: kite string, wooden alphabet blocks, and basic fasteners (see Figure 4). I hand out copies of the puzzle to groups of 4-5 students each, and ask them to partition or place the design.

Figure 4. Block and string model (left), with close-up of one block (right).

By having the physical model, students can quickly explore many different approaches to the problem. Many often discover iterative improvement techniques on their own, as well as develop cost models (number of nets cut by the partitioning, or wirelength) and clustering techniques. I have found the concrete, physical nature of the task motivates students, and reaches some students who often have trouble with more abstract formulations.

Figure 4(right) shows the current version of the model. I use wooden alphabet blocks, since they are cheap and have letters to designate individual nodes already. I use this to compare student efforts with my own solution. I drilled small holes into the blocks, then screwed in a screw eye fastener. In my original version of the puzzle I tied string directly to the eye fastener, but this meant it was very difficult to untangle the model after use. Instead, I use a binder ring (large metal circle), which can be opened and closed easily. To make the wires, I have taken lengths of kite string and added ends of ring terminal connectors (the yellow ends, crimped onto the string). All the parts can easily be obtained from toy, office supply, and hardware stores. This arrangement makes it easy to set up the puzzles for each offering, and have students untangle things when they work, by quickly slipping the strings on and off of the binder ring.

Conclusions

Incorporated active learning tasks tends to help promote student learning by having the students discover concepts on their own. However, how to incorporate active learning into specific engineering classes is not often obvious. In this paper I have covered multiple techniques for active learning in CAD. Most are easily replicatable at other institutions; the only complex system, Aphyds, is available from the author's website at the University of Washington. These techniques have proven to be very popular with students, and seems to significantly improve student engagement with the class.

References

[Hauck03] S. Hauck, "APHYDS: Academic Physical Design Skeleton", *IEEE Intl. Conf. on MSE*, 2003.