APHYDS: The Academic Physical Design Skeleton

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

Abstract

Physical Design is a complex CAD topic, which is difficult to teach to electrical and computer engineers. I have developed a complete skeleton for teaching basic algorithms such as Fiduccia-Mattheyses partitioning, variable node size sliceable floorplanning, simulated annealing placement, maze router based global routing, and left-edge algorithm channel routing. The Java-based toolset is portable, flexible, and provides a powerful interactive graphics interface.

Introduction

Today's electronics rely heavily on Computer-Aided Design (CAD) tools to support their design. As such, CAD has become a complex field, with a large amount of new, fundamental algorithms that need to be mastered. It also spans both electrical engineering and computer science; while electrical and computer engineers will need to understand how the tools work in order to understand how to use them effectively, much of the underlying theory and design requires fundamental concepts of computer science. As such, it is an important yet difficult subject to teach to undergraduates and graduate students.

When I joined the Department of Electrical Engineering in 1999 I began teaching a course on Physical Design, the portion of CAD after logic synthesis that concentrates on converting netlists into physical realizations. My first offering was based heavily on a course taught at Northwestern University by Majid Sarrafzadeh, which made use of the *DISPLAY* tool [1]. DISPLAY is a simple program that takes commands from a text file of simple primitives (rectangles, text, etc.) to display diagrams in X-windows. To use the tool for a physical design class, students are responsible for creating the entire system: this ranges from file reading and data structure development to writing complete complex algorithms, culminating in the emitting of a control file that DISPLAY uses to show the results.

After my first quarter of struggling with compiler bugs, compatibility issues, and the overall complexities of CAD development faced by electrical engineers in a 10-week class, I realized an alternative was needed. This paper describes the result: APHYDS, an integrated, Java physical design CAD toolsuite optimized for education purposes. It provides visualization and I/O routines, as well as basic data structures and interactivity, for the "canonical" standard algorithms in physical design. Students then focus strictly on implementing the aspects of the algorithms that are most important from a learning perspective. With the portability and interactivity of Java, and the guidance of a premade skeleton, the class has become significantly more effective and compelling.

Approach

My goal in developing Aphyds was to allow students who have a basic programming background to master multiple algorithms in a single academic quarter. Also, I felt that graphical display and interactivity was essential to a functioning CAD tool, and having a complete end-to-end system integrated together would be motivating for the students. A final design goal was to concentrate on fundamental, important algorithms from Physical Design, skipping the "latest and greatest" to focus on inculcating the background for future in-depth research.

In order to test-drive the system, and to help in development, an entire working toolsuite - including implementations of all algorithms students will eventually develop –was created over a 6 month period. Note that this also yielded a working system that has become a primary tool in lecture for demonstrating working algorithms, as well as a comparison target when students develop their own code. The system is approximately 14,800 lines of Java code. Special care was paid to parameter checking, assertion checking, and general error detection to make the system as bullet-proof as possible.

I examined each algorithm to identify the important subroutines, from an intellectual standpoint, which I then removed. Next, I comprehensively documented these procedures and the routines they required, as well as inserting additional diagnostic code around these routines.

The system was developed in Java for multiple reasons:

- Java is portable – students can develop code on PCs, Macs, UNIX, etc. without any code alterations. This includes portable graphics operations.
- The basics of Java are very close to C/C++, making it possible for students to learn the language on their own in a week or two.
- Java has good error detection & response facilities.

- Java's strong typing and memory model avoid common, pernicious bugs.
- Java's built-in documentation facility, JavaDoc, helps makes easily documented code.
- Java's iterators allow the encapsulation of complex data structure searches and other operations into simple objects.

Algorithms covered

Our goal is to introduce fundamental physical design algorithms, both to illustrate the constraints and to give students the background that is assumed by most CAD research efforts. These are organized in 6 programming assignments. Listed below are the assignment details, including the number of lines of code the students have to write (based on the solution I produced, including blank lines and comments) and the average amount of time it took to write in the most recent class offering.

Assignment 1: Learn Java and the basic Aphyds data structures by writing a topological sort to determine critical path length (43 lines, 15 hours).

Assignment 2: Fidducia-Mattheyses bipartitioning. Students produce a function to compute the single net gain function, and the FM main partitioning loop. The bucket data structure is provided to them. (113 lines, 20 hours).

Assignment 3: Floorplanning sliceable floorplans with variable node sizes. Students write subroutines to create vertically and horizontally split floorplans from sub-floorplans, and to merge the two lists. The slicing trees are pre-made. (119 lines, 13 hours).

Assignment 4: Simulated Annealing placement. Students write a semi-perimeter cost function, a move function, and the main simulated annealing loop. A semi-adaptive cooling schedule is provided, as is a greedy iterative improvement placer as a model for the annealer. (138 lines, 11 hours).

Assignment 5: Global routing via maze routing. Students develop a complete router. The routing tree data structure, and iterators to encapsulate the routing channel neighbor function are provided. (150 lines, 20 hours).

Assignment 6: Channel router. Students create a left-edge algorithm by inserting arcs into the vertical and horizontal constraint graphs, and write the main left-edge algorithm routine. The HCG and VCG data structures are provided. Unrouteable conflicts are avoided by moving some branches to polysilicon. (53 lines, 9 hours).

Classroom experiences

I have taught the class three times at the University of Washington – the first year with DISPLAY, and the following two with Aphyds. The fourth offering will be this winter quarter. The tool has been a significant success. While unscientific, trends in student evaluations and enrollment over the years are illuminating:

	99-00	00-01	01-02	02-03
Enrollment	23	13	24	32
Ave. Rating	4.5/5.0	4.8/5.0	4.8/5.0	---

The student evaluations are consistently higher for the APHYDS version of the class, and the enrollment numbers have been consistently rising since the introduction of APHYDS. There are a large number of complicating factors that make this data suspect, but it does provide interesting datapoints.

There are several things I have learned about the system from the last two years, some good and some not.

Aphyds advantages

- By handling the graphics and basic data structures, a much greater range of applications can be covered.
- Interactivity can be added to the tools, for realism and exploration. For example, facilities exist in Aphyds to allow students to move nodes by hand in the partitioning and placement phases, allowing students to explore.
- Tools can be augmented with statistics and graphs to illuminate concepts. For example graphs of cutsize during partitioning, and cost function during annealing, graphically expose how the programs work in practice.
- The completed instructor version becomes a compelling lecture aid. The tool can be run in class to show features of the covered algorithms.
- Continuous improvement has significantly improved the tool. A $50/homework bug-hunt in the first year exposed tool limitations and error checking omissions that I have corrected for future classes. Also, a redesign of an excessively hard task (global routing) via more intuitive data structures approximately halved student development time, bringing it in line with other assignments, and boosted success rates.

Aphyds disadvantages

- Integrating complex, unstructured final projects for a semester course is difficult, because students have no exposure to the GUI design necessary to add a new tool.
- The original class provided a programming "boot camp" for some students, forcing them to become better programmers due to the sheer bulk of work.

Conclusions

I have developed a toolsuite for teaching Physical Design CAD algorithms to electrical engineering students. It provides an infrastructure supporting portable graphics, and basic I/O and data structures, to allow students to focus on the essential, intellectual portions of the CAD algorithms. The system has been proven highly effective in two offerings of the class.

References

[1] M. Sarrafzadeh, C. K. Wong, *An Introduction to VLSI Physical Design*, McGraw-Hill, 1996