

California Institute of Technology  
Department of Computer Science  
Electronic Design Automation

CS137a, Winter 2002

Assignment #3

Monday, February 11

---

**Part A Due:** Friday, February 22, 11:59pm.

**Part B Due:** Friday, March 8, 11:59pm.

**Resources** You are free to use any books, articles, notes, or papers as references. Provide citations in your writeup as appropriate.

**Collaboration** Each student is expected to do his/her own work. For this project, you are free (and encouraged) to discuss basic strategies and approaches with your fellow classmates or others, but implementations, analysis, and writeups should always be the work of the individual. You may help each other with JAVA and HSRAppr API problems. If you get advice or insights from others that significantly influenced your work, please acknowledge this in your writeups.

**Writeup** Writeup should be in an electronically readable format (HTML or PDF preferred, Word tolerated). Turnin only the new classes you create (linking the java code to the writeup might be the best way to manage that). Code should be clearly commented. Writeup should indicate the role of each class you turn in and how they interact with the provided support code.

## Problems

**Given:** A mapped netlist of LUTs (could be any bounded-input, single output netlist) and IOs.

**Goal:** Provide a placement of LUTs onto a 2D grid minimizing critical path delay measured in total manhattan distance between registers or between registers, inputs, and outputs. (To simplify the problem you do not have to place IOs, you can assume zero cost to travel from a primary input or to a primary output.)

**Part A:**

1. Develop a placer which randomly places the LUTs onto the array; collect and report on the solution quality in terms of the aforementioned objective. Compare VPR placements to random placements.
2. Sketch pseudocode for the algorithm you plan to use to solve the problem.

**Part B:**

1. Develop an implementation for your optimizing placer. Benchmark it versus the random placer and VPR.
2. Writeup should include:
  - (a) Basic problem formulation (including optimization criteria and cost model)
  - (b) Outline of solution, including algorithm description (could be same as on Part A, but I suspect it will evolve as you get feedback from implementation)
  - (c) Instructions on how to run your optimizer (arguments, options, etc.)
  - (d) Results and comparison (benchmark results, etc.)
  - (e) Lessons and recommendations including:
    - important issues and lessons in the design and implementation of the algorithm;
    - discussion of the optimality of the solutions produced by this algorithm and implementation;
    - discussion of the next set of experiments and studies which would be beneficial to further improve this placer.

## Additional Information

Base for support code and examples are in `/cs/courses/cs137/assign3/`. You'll want to make your own copy of the `copy_this` directory as a starting point. It includes:

- `ENVIRONMENT` which you will need to modify appropriately (and you may want to adapt it to your shell)
- `dummy_place.java` to use as a base driver for optimization
- `Dummy2DPlace.java` sample placement class
- `Makefile` for running java compiler plus an example invocation of the existing `dummy_place` once built.

You may want to grab other classes from `HSRAppr` to modify according to your needs.

Examples live in the `netlists` subdirectory of the assignment directory.

The `dummy_place` driver provides a suitable main routine and sample driver for you to start with. It reads in `netlists` and calls `minCycle` and `maxDistance` with a suitable distance cost function (`TimingDomains2D`) to analyze a placement. You should be able to copy this and replace the dummy placer given with your randomizing and optimizing placers for this assignment.

Source for `HSRAppr.jar` lives in `/cs/research/ic/develop/HSRAppr/src/`.

For benchmark reporting for this assignment, use the 6 examples contained in `netlists`. Rafi has produced VPR placements for these in `vpr_place`. You can run `dd2d` to read in existing placements (such as these) and analyze their quality; the provided `Makefile` has a rule to run `dd2d`.

Rafi has experience using working with the `HSRAppr` suite and will also be able to answer your questions about it.