EECS 527 - Homework 3

Topic: Empirical Evaluation of an Analytical Placement Algorithm

Optimization problem and objective functions:

- linear ordering with total span and max-cut, same as in mini-project 1

Main action items:

- Learn how to run the Capo placer to produce linear orderings
- Implement a simple analytical algorithm for linear placement (described below)
- Compare the performance of the two with the best algorithm from mini-project 1 (BFS)
- Analyze empirical results and explain your findings

Hand in:

- for BFS and the simple analytical algorithm, the number of lines of code (LOC) in your implementation (try to minimize #LOC)
- for each of the 3 algorithms and for each of 18 IBM benchmarks,
  * average results per run (span, max-cut, runtime, memory consumption)
  * best seen solution qualities (to make a fair comparison, adjust the number of runs for each algorithm to equalize total time)
- analysis of empirical results: half a page to a page of text

Download site for IBM circuit benchmarks:
http://vlsicad.cs.ucla.edu/~cheese/ispd98.html

Download site for UCLA Physical Design Tools (UCLA pack) which include Capo
http://vlsicad.eecs.umich.edu/software/PDtools/

What to do:

- Use the same UCLApack distribution as in mini-project 2. Check that the MetaPlacer package was installed successfuly. For that, first check that the file MetaPlacer/MetaPlacerTest0.exe exists. Then run the script regression in the MetaPlacer package. The script may produce results that are slightly different from pre-computed results, but at least there should be no segmentation faults or crashes.
- To run the Capo placer, use the MetaPlacerTest0.exe executable in the MetaPlacer package (it runs Capo and postprocesses the results using optimal placers). See how the regression script does that
- Prepare .aux files, one for each IBM benchark, similar to TESTS/fadi1.aux (when Capo/MetaPlacer sees such an aux file, it runs in the linear placement mode; it can use either nets/nodes or net/are files)
- Implement the following linear placement algorithm:
  1) start with a random placement (or use BFS, if you like);
  2) for each hyperedge, compute the average location of all pins, call it the center of gravity (COG) of the hyperedge;
  3) for each vertex, compute the average location of COGs of all incident hypergedes
  4) sort() vertices based on those average locations and re-order/re-place them based on sorted indices
  5) compute total span and max-cut for the new ordering
  6) repeat items 2)-5) until the objective functions improve

  Note: make sure to call STL's sort function with an inline comparison operator on an array (rather than a linked list, etc)
- Compare Capo, your analytical algorithm and the BFS-based algorithm

from MiniProj 2 by running them on IBM benchmarks

- Describe your findings