University of Toronto, Faculty of Applied Science and Engineering
Department of Electrical and Computer Engineering

ECE 1387 - CAD for Digital Circuit Synthesis and Layout
Assignment #2 – Analytical Placement with Heterogeneous Cell Types, Spreading, Snapping/Legalizing to Grid

October 2018 J. Anderson

Assignment Date: October 14, 2018
Due Date: November 2, 2018, at the beginning of class.
Late Penalty: **-2 marks per day late, with total marks available = 20**

You are to write an implementation of a basic analytical placer (AP), with overlap removal (spreading) and fitting/snapping-to-grid. As described in class, you will formulate the placement problem mathematically as a system of linear equations to be solved. You will use an existing package (UMFPACK) to solve the linear system (see announcement on Piazza page). Your placer will also handle heterogeneous block (cell) types, where there are restrictions on where certain types of cells may be placed. Such heterogeneity arises in FPGA placement, where for example, DSP and RAM blocks can only be placed in specific locations on the device.

Your program should display its progress and results using the same graphics package as used in Assignment #1 (available on course webpage). Your graphics should show the placement results and the connectivity between blocks (rat's nest of wires). Blocks (cells) should appear as points in your placement and be labeled with block numbers (see below). *There is an example executable from a prior year of ECE1387 on the Piazza site, if you'd like some inspiration for how the placement may be displayed.*

The netlist file input format has two sections. The two sections are separated from one another by a –1 appearing by itself on a line. The first section specifies the blocks to be placed and the connectivity between them. Each line has the following form:

blocknum type $netnum_1$ $netnum_2$ $netnum_3$ ... $netnum_n$ -1

Where blocknum is a positive integer giving the number of the cell, type is either 0 or 1 (the type of the block), and the $netnum_i$ are the numbers of the nets that are attached to that block. Every block that has the same $netnum_i$ on its description line is attached. Note that each block may have a different number of nets attached to it. Each line is terminated by a –1.

Example input file:

```
1 0 2 3 4 -1
2 1 5 4 -1
3 0 5 6 2 -1
4 0 6 3 -1
 -1
1 50 0
4 0 50
-1
```

In this example, block 1 (of type 0) is connected to nets 2, 3 and 4. Note that each net may be connected to more than two blocks (that is, there are multi-fanout nets). Also note that net numbers are not related to block numbers.

As discussed in class, the AP formulation requires there to be a set of pre-placed (fixed) objects, normally I/Os. The second section of the netlist file specifies the placement of fixed objects. It has the following form:

blocknum x_position y_position

In the above example, block 1 is pre-placed at the position with x = 50, y = 0. The list of fixed objects is terminated by a –1 by itself on a line.

You should run your placer on the Assignment #2 test circuits provided on the course web page.

Each test circuit is to be placed on an NxN grid of slots (each slot is 1 area-unit wide and 1 area-unit tall), where N = ceiling(sqrt(m)), and where m is the # of blocks. m = #fixed_blocks + #moveable_blocks. The idea is that each cell/block is to eventually be fit into one such slot.

What to do and what to hand in?

Your placer must run on the ECF network. Instructions for electronic submission of your placer (including source code) will be posted on the course's Piazza page close to the assignment deadline.

Your report should include a short description of the flow of your program, the main routines and what they do, assuming that I have basic knowledge of analytical placement.

1. Formulate and solve the analytical placement problem assuming the *clique* net model[1]. Do not do any overlap removal in this step. Your program should display the placement and rat's nest (wires between cells) using the graphics package. Hand in a plot of the placement results. Your program should also compute the half-perimeter bounding box (BB) wirelength (WL) of the placement. Hand in a table showing the half-perimeter BB WL for each placed test circuit.

2. **For circuits 2-4 only:** Implement a simple form of overlap removal. Given that the placement area spans from (0,0) to (N,N), use the results of Step 1 to divide the non-fixed blocks into four groups of roughly equal size. Each group will consist of the blocks closest to each quadrant of the placement area. That is, one group will represent the ~n/4 blocks closest to the lower-left corner of the placement; a second group will represent the ~n/4 blocks closest to the top-left corner of the placement, and so on. Having divided the blocks, reformulate the placement problem as in Step 1, but with a modification: In your reformulation, introduce 4 *new* (artificial) fixed blocks, placed at the **four centres** of the die quadrants (i.e., one at (N/4, N/4), one at (N/4, 3N/4), etc.). Introduce (artificial) two-pin nets from each new fixed block to each block in the group corresponding to the fixed block's quadrant. Solve the formulation, compute its BB wirelength, and display and plot the results. Describe how you partitioned the blocks. Hand in a table showing the BB WLs. Do not include the WL of the artificial connections in your WL numbers. What happens to WL when the new fixed blocks are introduced? Experiment with changing (increasing/decreasing) the weights of the artificial two-pin nets; comment on the WL results when different weights are used.

3. **For circuits 2-4 only:** Repeat step #2 recursively to perform further spreading by continuing to modify the mathematical formulation. For example, take the group of blocks you assigned to the lower-left quadrant in Step #2, and divide this group into four sub-groups, each corresponding to a sub-quadrant of the lower-left quadrant. Introduce fixed blocks at the centers of the four sub-quadrants. Introduce (artificial) two-pin nets from each new fixed block to each block in the corresponding sub-group. Do the same for the other quadrants of the die. Solve the new placement formulation. Repeat the process recursively, with successively smaller regions. I suggest you continue spreading until no more than 15%

[1] In the clique model, a net with p pins is represented as a complete graph (clique) with p*(p-1)/2 edges. Each edge in the complete graph has weight of 2/p. For example, a net with 2 pins has 1 edge with edge weight = 1. A net with 4 pins has 6 edges with edge weight = 2/4.

of 1x1 slots are overfilled. Provide a plot of the spread placement. In a table, report the BB WL when your stopping criterion is met.

4. **For circuits 2-4 only:** "Snap"/legalize the placement produced by step #3 into the grid, where cells are points placed at the centre of the 1x1 grid slots, and there is *at most* one cell placed in each such grid slot.

   You are to do this in two ways: 1) where any type of block can be placed anywhere, and 2) where type 1 blocks can only be placed in every fifth column (beginning from the second column from the left), as illustrated in the 12x12 (N=12) example figure below.

   For both fitting approaches, provide a plot of the "snapped" placement for each circuit. Likewise, for both fitting approaches, report the BB WL after snapping and compare with the results for step #3. How much did BB WL increase after snapping? How much is the BB WL affected by the placement restrictions for type 1? Time permitting, you may consider innovative ideas to adjust your spreading in steps 2 and 3 above to be "aware" of the block types and their potential placement locations. **In class, I will report on the results achieved by everybody and mention the best WL achieved.**

OPTIONAL: Instead of applying step #3 recursively as above, you may use ANY spreading strategy you wish, including SimPL, as described in class. If you decide to go this route, briefly describe your solution and optimizations. Note that it is not my expectation (or a requirement for full marks) that everyone opt for this route and implement SimPL or some other more advanced algorithm.