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 and Spreading

March, 2010 J. Anderson

Assignment Date: March 3, 2010
Due Date: March 17, 2010, at the beginning of class.
Late Penalty: **-1 marks per day late, with total marks available = 10**

You are to write an implementation of an analytical placer (AP) with overlap removal (spreading). 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 represent the matrices and vectors, and then to solve the linear system (available on course web page).

Your program should display its progress and results using X11-based graphics, as in Assignment #1, using the same graphics package. 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.

The netlist file input format has three sections. The three sections are separated from one another by a –1 appearing by itself on a line. The first section has only a single line X Y, indicating the extent of the placement region in the X and Y dimensions. The second section specifies the blocks to be placed and the connectivity between them. Each line has the following form:

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

where blkname is the name of the cell, and blocknum is a positive integer giving the number of the cell, 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:

```
50 50
-1
blk1 1 2 3 4 -1
blk2 2 5 4 -1
blk3 3 5 6 2 -1
blk4 4 6 3 -1
 -1
blk1 1 50 0
blk4 4 0 50
-1
```

The first line shows that the placement region spans 50 units in the X and Y dimensions. Moving onto the second section, observe that block 1 (called "blk1") 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) cells, usually I/Os. The third section of the netlist file specifies the placement of fixed cells. It has the following form:

blkname 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 cells is terminated by a –1 by itself on a line.

You should run your placer on the **six** test circuits provided on the course web page. These are real circuits from the MCNC circuit benchmark suite.

What to do and what to hand in?

Hand in a listing of your program and the location of the executable. Provide instructions on how to run your program. Make sure to set file and directory permissions so I can access your placer.

You should hand in 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. [4 marks] Formulate and solve the analytical placement problem. Use 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 wirelength (HPWL) of the placement. Hand in a table showing the HPWL for each placed circuit.

NOTE: You will need to read Section 2 of the UMFPACK quick start guide (on the web page) to formulate the problem in UMFPACK's sparse matrix format and then solve the system.

2. [2 marks] **Do this for circuit tseng.ap ONLY:** Examine the impact of I/O placement on the placement results. Assuming the same set of fixed locations, have your program randomly interchange some of the fixed blocks with one another and repeat Step #1 above for each different I/O placement. Find and report the average, minimum, and maximum HPWLs across a number of random I/O placements (try enough so the average doesn't change much).

3. [4 marks] Implement overlap removal. Given that the placement area spans from (0,0) to (X,Y), assume that the area is partitioned into a set of X*Y regions, *R*, each with dimensions of 1x1 units, continue spreading the moveable (core logic) cells until the following stopping criterion is met:

$$Overlap = \frac{\sum_{r \in R} \max(0, m_r - 1)}{m} \leq 0.15$$

where m is the total number of *moveable* core logic cells (non-fixed cells), and m_r is the number of *moveable* cells placed in region r. m_r should be computed assuming that cells are points with zero area. The intuition behind the above relation is that each 1x1 region can accommodate one core logic cell. You should aim to meet the above criterion with minimal damage to HPWL. **For the bigkey.ap circuit only:** provide plots that show the progression of the spreading and provide a plot of the spread placement that meets the stopping criteria. **For all test circuits:** Report the HPWL when the stopping criterion is met. In the report, briefly describe your implementation and highlight any clever techniques you used. *You are free to implement any overlap removal scheme.* You are welcome to use a simplified "FastPlace-like" overlap removal, as described in class and in the FastPlace paper. Note that in FastPlace, your placement bin structure (for cell shifting) does not need to match with the regions used for calculating overlap. If you use pseudo-pins and nets, do not include the lengths of pseudo nets in your HPWL values! **In class, I will report on the various techniques everybody attempted, and the best results achieved.**

[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.