Physical Design for Nanometer ICs Programming Assignment 3

B07901087 電機三 蔡沛叵

1. File explanation:

This program is written in C++, and there are several cpp and header file for my floorplanner.

GlobalPlacer.h: some parameters for global placer.

GlobalPlacer.cpp: placement initialization, call analytical solver to solve objective function.

ExampleFunction.h: some parameters for WL and density models,

ExampleFunction.cpp: implementations for WL/Density models

To compile the binary, please see readme.txt

2. data structure: (some common parameters)

int _binRow; number of bin in total placement area in row direction

int _binCol; number of bin in total placement area in column direction

int _binNum; total number of bin in placement area

double _binW; bin's width

double _binH; bin's height

double _alpha; alpha in sigmoid function

double _currlambda; lambda in sigmoid density function

double _initWL; initial wirelength used to normalize WL's f

double _gamma; gamma in LSE model

the following four parameter are used to modify f, g part of WL and density models

double _fWLReviseGlobal;

double _gWLReviseGlobal;

double _fDenseReviseGlobal;

double _gDenseReviseGlobal;

3.Algorithm:

Grid:

Divide the total placement area into Row * Col grids.

WL model:

Use LSE wirelength model which was taught in class

Density model:

Use sigmoid function taught in class, however, since my alpha is larger, so I only need to check module within a bounding box when computing module density in a box.

4.Discussion:

Initial Solution:

To ensure a legal solution at the beginning, I randomly allocate modules into 15*15 bins uniformly to make sure that the placement result is sparse enough to be legalized.

I also implemented another initialization method. Due to the observation that all modules have low degree, I thought that maybe all modules are loosely connected, then I can simply sort nets by their degree in decreasing order and place corresponding modules of nets in the same grid. However, to my surprise, this didn't make the WL improve a lot(only improve densely connected cases), and legalization becomes harder. I still don't know the answer… maybe there are some defects in my implementation?

Density function:

我觀察最多的應該是 alpha 對於 density 的影響，當 alpha 小的時候，sigmoid density function 就會變得比較沒有作用，即使 module 隔了 2 個 bin 可能也就從 0.3 降到 0.2 幾…但是只要 alpha 一調大就可以讓不同 bin 的 module 在這個 bin 的 density 快速下降(例如: alpha = 0.01 的時候差不多隔一個 bin 就會差個 10 倍左右)，所以 alpha 應該算是控制 bin density 敏感度的參數，但是由於動一點就會導致結果變化很多，其實還蠻難調整的…，另外，由於外圍的 bin 都少了一側的 module，所以 density 會比較小，導致 module 會被吸過去，最後向外噴出…所以只要 density function 一調大就會發現多出一大堆 outlier。

我最後 density function 應該還是沒有調整好…只要一把它弄到和 WL 差不多，兩個 function 就會開始打架，gradient 就不會動，但是如果不 normalize 的話，density function 就會輸給 WL，導致最後一步的 legalization 變得困難

另外，為了加速，我有使用林予康同學介紹的 Fast exponential

ref:

https://stackoverflow.com/questions/47025373/fastest-implementation-of-the-natural-exponential-function-using-sse