EE 5301 - Fall 2018

Project 3: Simulated Annealing Non-Slicing Floorplanning

Due dates:
Phase 1: Dec 4, 2018 by 11:59pm
Phase 2: Dec 18, 2018 by 11:59pm

This programming assignment requires you to use C/C++ to implement an annealing-based non-slicing floorplanning on a new set of benchmarks. You will use the sequence-pair data structure to implement the floorplanning algorithm. The cost function is a linear combination of the total chip area and the sum of half-perimeter wire lengths (HPWL). To reduce the programming effort, the file format has been simplified, and all blocks are restricted to have only one shape with no rotation allowed.

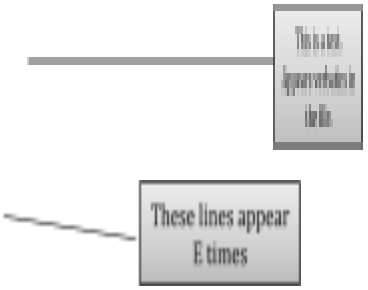
Wire length is defined as the sum of all the half-perimeter wire lengths of all the **hyper-nets**. The half-perimeter wire length (HPWL) of a hyper-net is half the perimeter of the smallest rectangle that encompasses all the **center points** of the blocks connected by that hyper-net.

You are supposed to tune your annealing parameters to get good area and wire length results. The parameters include K, the initial temperature, the freezing temperature, the number of moves per temperature step and the cooling schedule.

Input File Format

To show the file format, we use the notation `<some description>`, which refers to an *integer* that is described by the text `some description`.

```
<N = number of blocks>
<block number 0> <width> <height>
<block number 1> <width> <height>
. . .
<block number N-1> <width> <height>
Nets
<number of hyper-edges>
<degree of hyper-edge> <block> ... <block>
<degree of hyper-edge> <block> ... <block>
<degree of hyper-edge> <block> ... <block>
. . .
```



As an example, a toy file is shown below with added comments in red ink :

```
4                   number of blocks
0  10  20           Block  0, with the shape 10x20 (width x height)
1  5  5             Block 1, shape 5x5
2  3  8
3  9  8             Block 4
Nets
3                   3 hyper-edges to follow
2  3  1             degree=2, blocks connected are 3 and 1
4  2  0  1  3       degree=4, blocks connected are 2, 0, 1, 3
2  0  1             degree=2, block connected are 0 and 1
```

Phase 1

In Phase 1, you should build the backbone of the program: reading the input file, populating the data structures, generating a random initial floorplan and calculating the coordinates of the bottom-left corners of all the blocks based on the random initial floorplan. No wire length calculation, annealing moves or annealing engine are needed, but you do need to do the longest path to find module coordinates.

The program takes one command-line parameter, which is the name of the input file. You should output the sequence pair, chip width and height, the area of the chip, and the bottom-left corners of all the blocks to a file with the name `InFileName_YourLastName_YourFirstName.out1`.

Phase 2

In Phase 2, you would implement the annealing engine, along with the move set, and wire length calculations. You will experiment with annealing parameters to fine-tune your placement engine to get better solution convergence. Your report should highlight your strategy and the thinking that went into tuning your code. You will report your area / wire length results for the given benchmarks. Part of your grade will be determined by the quality of your floorplans. I expect to see some innovation in this phase (e.g., optimizing the annealing engine, analyzing annealing behavior when some parameters change, etc.).

The program has to take two arguments: the first one is the input file name (as in Phase 1), and the second argument is either "-a", "-w", or "-c" without the quotation marks. The **"-a"** option indicates that the objective is only to minimize area, disregarding the wire length. The **"-w"** options means the objective is only wire length minimization, disregarding area, and the **"-c"** options means a combination of area and wire length is the objective.

When implementing your code, your cost function would be of the form $\propto$ area + (1- $\propto$) HPWL, in which "area" is the area of the chip, and "HPWL" is the wire length

cost. When the program is run with the "-a" option, then you set $\propto = 1$. For "-w", you set $\propto = 0$ in the program. You are supposed to experimentally find the best $\propto$ value that works best for minimizing both wire length and area. The results in this mode probably have worse wire length, but better area compared to the –w mode.

Phase 2 Output Format

The output file name should be

`InFileName_YourLastName_YourFirstName.out2a` if –a was used
`InFileName_YourLastName_YourFirstName.out2w` if –w was used
`InFileName_YourLastName_YourFirstName.out2c` if –c was used

The output file format (ASCII, not binary) for each circuit should be as follows:

- The total HPWL of the placement (regardless of the command-line argument used).
- The width, height, and the total area of the placement (regardless of the command-line argument used).
- |V| lines, each stating the x y coordinate of a block.

You should also create one pdf file containing the results table. The table should look like this:

Ckt name	Argument	Chip Width	Chip Height	HPWL	Runtime
n10	-a	...	...	...	...
n10	-w				
n10	-c				
n30	-a	...			
...					

Below the table, show a bullet list of your strategy and any interesting techniques you used or observations you made during the assignment.

The benchmarks are posted on Canvas.

The source of the benchmarks is:
http://vlsicad.eecs.umich.edu/BK/GSRCbench/HARD/

Please note that the format has been modified and I/O pins removed to simplify it for you to work on.

Submission Process

- In each phase, please submit your code as one .zip containing all source and header files, and the output files for each of the circuits in assignment 1-b.
- Also in Phase 2, include the pdf file with the results table and your strategy/observations. The name of the zip file should be YourUmnId.zip.
- Please submit a readme to explain your implementation, approach, any issues you faced and the considerations to make when running your code.
- A Makefile to compile your code to a single executable is also required to be submitted.
- Please make sure your code runs on the CSE Lab machines. This check helps to understand if you need to include any compilation flags in your Makefile to run on the environment, that the Lab machines support.

Grading

- Submission and compilation of the program: 25%
- Legal floorplans (no block overlaps): 10%
- Wire length quality in the –w option: 10% (the worst wire length quality in class gets 5%, and the best gets 10%, and the rest are linearly graded in between). Wire length quality is the arithmetic mean of the HPWL of all the circuits in mode -w.
- Area quality in the –a option: 10% (the worst area quality in class gets 5%, and the best gets 10%, and the rest are linearly graded in between).
- Combined area and wire-length quality in the –c option: 10% (the worst cost quality in class gets 5%, and the best gets 10%, and the rest are linearly graded in between). The cost is calculated as (area+wirelength), and averaged over all benchmarks.
- Runtime of the –c option: 20% (the slowest code will get 5%, the fastest will get 20%, and the rest linearly in between).
- Documentation, including the results table and your strategy/observations: 15%