CS6135 VLSI Physical Design Automation

Homework 3

Fixed-outline Floorplanning with Fixed and Soft Modules

Due: 23:59, November 26, 2023

1. Introduction

Given a fixed outline, a set of rectangular hard modules each with a fixed position, a set of rectangular soft modules each with a minimum area, and a set of 2-pin nets along with their weights, you are asked to implement an existing algorithm or develop your own algorithm to determine the shape and position of each soft module. All modules must be placed within the outline without overlapping, with the objective of minimizing the total weighted wirelength between modules.

2. Problem Description

(1) Input:

- ✓ A chip with known width and height
- ✓ A set of soft modules and their minimum areas
- ✓ A set of fixed modules and their positions (i.e., the coordinates of their lower-left corners), widths, and heights
- ✓ A set of 2-pin nets and their weights

(2) Output:

- ✓ The total weighted wirelength
- ✓ The position, width, and height for each soft module

(3) Constraints:

- ✓ Fixed modules must be placed at their given positions.
- ✓ Soft modules must be rectangles.
- ✓ Soft modules must be within the chip.
- ✓ The aspect ratio (height/width) of each soft module must be between 0.5 and 2.
- ✓ The width, height, and coordinates of each soft module must be integers.
- ✓ The area of each soft module must be no less than its minimum area.
- ✓ There must be no overlap between modules.

(4) Objective:

- ✓ The shorter the total weighted wirelength, the better.
- ✓ The weighted wirelength for each net is defined as the product of the specified weight and the half-perimeter wirelength (HPWL) of the minimum bounding box of pins of the net. Each pin of module m_i is located at the center of m_i. Note that the x- or y- coordinate, say i, of the center of each module is rounded down to an integer k such that

$k \leq i < k + 1$.

3. Input File

(1) The *.txt* file:

The .txt file specifies the input information. Here is an example:

```
ChipSize 8 7
// ChipSize chip width chip height

NumSoftModules 2
// NumSoftModules number of soft modules
SoftModule GPU 25
// SoftModule module name minimum area
      ⋮
NumFixedModules 2
// NumFixedModules number of fixed modules
FixedModule PAD1 0 5 2 5
// FixedModule module name x coordinate y coordinate module width module height
      ⋮
NumNets 3
// NumNets number of nets
Net GPU CPU 20
// Net module1 name module2 name net weight
      ⋮
```

4. Output File

(1) The *.floorplan* file:

The .floorplan file specifies the floorplanning result including the total weighted wirelength of all nets, and the coordinates of the lower-left corner of each soft module and its width and height.

```
Wirelength 215
// Wirelength wirelength

NumSoftModules 2
// NumSoftModules number of soft modules
GPU 0 0 5 5
// module name x coordinate y coordinate module width module height
      ⋮
```

5. Language/Platform

(1) Language: C/C++

(2) Platform: Unix/Linux

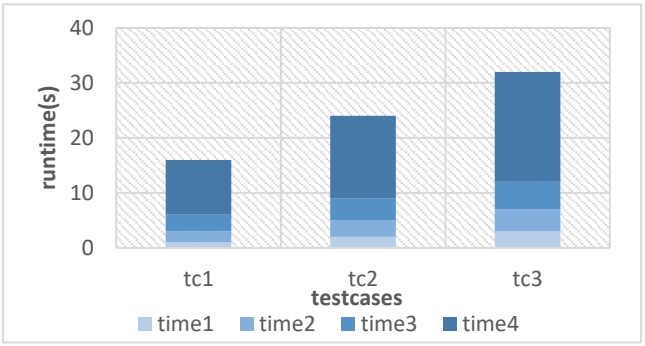
6. Report

Your report must contain the following contents, and you can add more as you wish.

(1) Your name and student ID

(2) How to compile and execute your program and give an execution example.

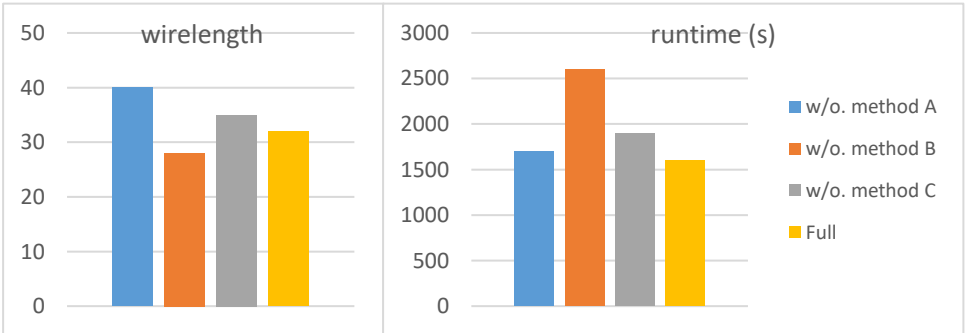
(3) The wirelength and the runtime of each testcase. Notice that the runtime contains I/O, constructing data structures, initial floorplanning, computing parts, etc. The more details your experiments have, the more clearly you will know where the runtime bottlenecks are. You can plot your results like the one shown below.



(4) How did you determine the shapes of the soft modules? What are the benefits of your approach?

(5) The details of your floorplanning algorithm. You could use flow chart(s) and/or pseudo code to help elaborate your algorithm. If your algorithm is similar to some previous work, please cite the corresponding paper(s) and reveal your difference(s).

(6) Try your best to enhance your solution quality. What tricks did you do to enhance your solution quality? Also plot the effects of those different settings like the ones shown below.



(7) If you implement parallelization (for algorithm itself), please describe the implementation details and provide some experimental results.

(8) What have you learned from this homework? What problem(s) have you encountered in this homework?

7. Required Items

Please compress `HW3/` (using tar) into one with the name `CS6135_HW3_${StudentID}.tar.gz` before uploading it to eeclass.

(1) `src/` contains all your source code, your `Makefile` and `README`.
   - `README` must contain how to compile and execute your program. An example is like the one shown in HW2.

(2) `output/` contains all your outputs of testcases for TAs to verify.

(3) `bin/` contains your executable file.

(4) `CS6135_HW3_${STUDENT_ID}_report.pdf` contains your report.

You can use the following command to compress your directory on a workstation:

```
$ tar -zcvf CS6135_HW3_{StudentID}.tar.gz <directory>
```

For example:

```
$ tar -zcvf CS6135_HW3_112062500.tar.gz HW3/
```

8. Grading

- ✓ 80%: The solution quality (wirelength) of each testcase, hidden testcases included. This part will be evaluated with the sequential version of your program.
- ✓ 20%: The completeness of your report
- ✓ **5% Bonus**: Parallelization

Notes:

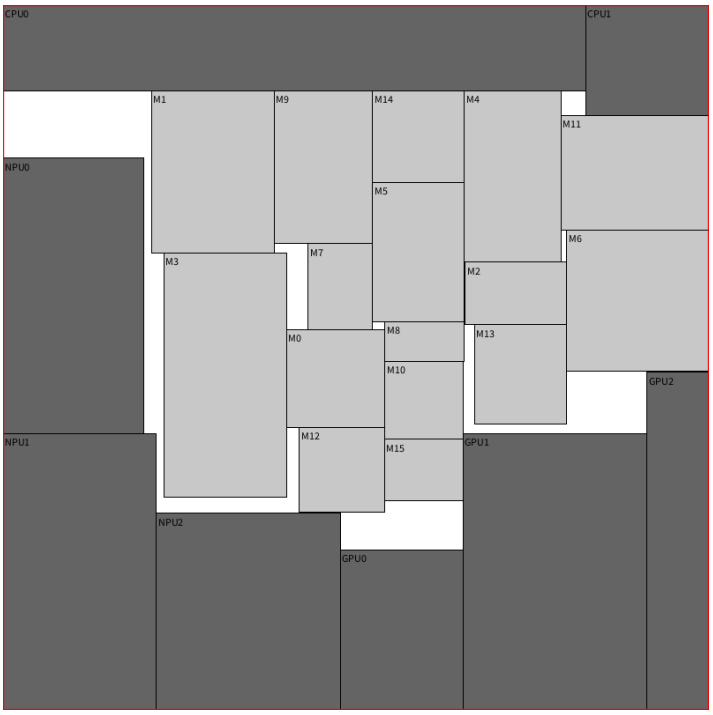
- Make sure the following commands can be executed.
  - Go into directory "`src/`", enter "`make`" to compile your program and generate the executable file, called "`hw3`", which will be in directory "`bin/`".
  - Go into directory "`src/`", enter "`make clean`" to delete your executable file.
- Please use the following command format to run your program.

  ```
  $ ./hw3 *.txt *.floorplan
  ```

  E.g.:

  ```
  $ ./hw3 ../testcase/public1.txt ../output/public1.floorplan
  ```
- If you implement parallelization, please name the executable file of your parallel version as "hw3_parallel" and name the executable file of your sequential version as "hw3".
- Use arguments to read the file path. Do not write the file path in your code.

- Your program must be terminated within 10 minutes for each testcase.
- We will test your program by a shell script with GCC 9.3.0 on ic51. Please make sure your program can be executed by **HW3_grading.sh**. If we cannot compile or execute your program by the script, you will get 0 points on your programming score.
- For each testcase, you could use **HW3_printer** to draw your result like the following figure, where each black rectangle is a fixed module and each gray rectangle is a soft module.



- Note that any form of plagiarism is strictly prohibited. If you have any question about the homework, please contact TAs. (If you have any question about C/C++ programming, please google it by yourself.)