CS6135 VLSI Physical Design Automation

Homework 4: Global Placement

Due: 23:59, December 17, 2023

1. Problem Description

This programming assignment asks you to write a global placer that can assign modules to desired positions on a chip. Given a set of modules, a set of nets, and a set of pins for each module, the global placer places all modules within a rectangular chip. In the global placement stage, overlaps between modules are allowed; however, modules are expected to be distributed appropriately such that they can easily be legalized (placed without any overlaps) in the later stage. As illustrated in Figure 1, for a global placement result with modules not distributed appropriately (Figure 1(a)), the modules cannot be legalized as illustrated in Figure 1(b) (modules in the middle bin of Figure 1(b) are overlapped). In Figure 1(c), a more appropriately distributed global placement result is provided, which can be legalized as illustrated in Figure 1(d).

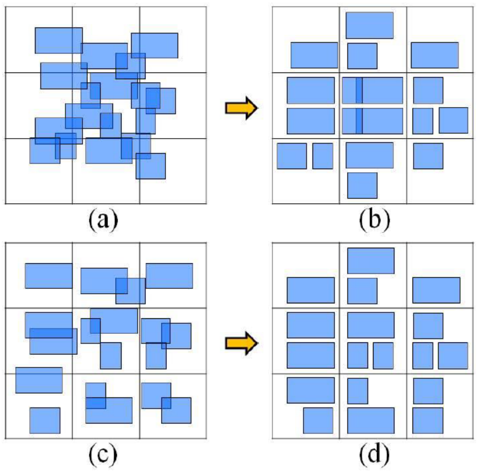

Figure 1. The process of global placement and legalization

In addition to placing modules appropriately, the objective of global placement is to minimize the total net wirelength. The total wirelength W of a set N of nets can be computed by

$$W = \sum_{n_i \in N} HPWL(n_i)$$

where n_i denotes a net in N, and $HPWL(n_i)$ denotes the half-perimeter wirelength of n_i. Note that a global placement result which cannot be legalized is not acceptable, and any module placed out of the chip boundary would lead to a failed result.

2. Predefined Data Structure

```
class Module {
  public:
    /* get functions */
    const char* name();
    double x(); // coordinate of the lower-left corner
    double y();
    double width();
    double height();
    bool isFixed(); // whether the module is fixed or not
    double centerX();
    double centerY();
    double area();
    const char* orientString();
    unsigned numPins();
    Pin& pin(unsigned index); // index: 0 ~ numPins()-1

    /* set functions */
    // use this function to set the module lower-left position
    void setPosition(double x, double y);
};

class Net {
  public:
    unsigned numPins();
    Pin& pin(unsigned index); // index: 0 ~ numPins()-1
};

class Pin {
  public:
    double x();
    double y();
    unsigned moduleId();
    unsigned netId();
};
```

```
class Placement {
  public:
    const char* name();
    const char* plname();
    double boundaryTop();
    double boundaryLeft();
    double boundaryBottom();
    double boundaryRight();
    Module& module(unsigned moduleId);
    Net& net(unsigned netId);
    Pin& pin(unsigned pinId);
    unsigned numModules();
    unsigned numNets();
    unsigned numPins();
    double computeHpwl(); // compute total wirelength
    double computeTotalNetLength(int cellid);

    // output global placement result file for later stages
    void outputBookshelfFormat(const char *const filePathName);
};
```

The above functions are used to access the data required by the global placement stage. You should use the function `setPosition(double x, double y)` to update the lower-left coordinates of moveable modules (whose positions are to be determined). You can use `isFixed()` to check whether the module is fixed or not. At the same time, the coordinates of pins on modules will be updated accordingly and automatically. Note that modules cannot be placed out of the chip boundaries, or the program may not be executed normally.

3. Output Format

If you didn't use the provided code and implement your own program, please make sure the output format corresponds to the following:

```
UCLA pl 1.0

a0 -27197 7830 : N
// module name x coordinate y coordinate : orientation
      ⋮
```

4. Language/Platform

(1) Language: C/C++

(2) Platform: Unix/Linux

5. Report

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

(1) Your name and student ID

(2) The wirelength and the runtime of each testcase

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

(4) Try your best to enhance your solution quality. What tricks did you do to enhance your solution quality?

(5) Please compare your results with the previous top 5 students' results and show your advantage in solution quality. Are your results better than theirs?

- ✓ If so, please express your advantages to beat them.
- ✓ If not, it's fine. If your solution quality is inferior, what do you think that you could do to improve the result in the future?

Previous top 5 students' results

	Wirelength		
Rank	public1	public2	public3
1	68783367	8778312	456197589
2	79542407	9155930	478967869
3	83860394	9783498	463664700
4	73804994	11105419	413161709
5	80176617	10921603	539864787

6. Required Items

Please compress `HW4/` (using tar) into one with the name `CS6135_HW4_${StudentID}.tar.gz` before uploading it to eeclass.

(1) `src/` contains all your source code.

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

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

(4) `CS6135_HW4_${StudentID}_report.pdf` contains your report.

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

```
$ tar -zcvf CS6135_HW4_${StudentID}.tar.gz <directory>
```

For example:

```
$ tar -zcvf CS6135_HW4_112062500.tar.gz HW4/
```

7. Grading

- ✓ 80%: The solution quality (wirelength) of each testcase, hidden testcases included.
- ✓ 20%: The completeness of your report.

8. Evaluation

If the global placement result can be legalized, the final solution quality is judged by the total HPWL after the detailed placement stage, which will be shown on the screen as follows:

```
Benchmark: public1


Global HPWL: 729931675
 Legal HPWL: 684601696   Time:    0.42 sec (0.01 min)
Detail HPWL: 324193780   Time:    4.08 sec (0.07 min)
 ==================================================================
       HPWL: 324193780   Time:    4.50 sec (0.07 min)


[Success] Your output file satisfies our basic requirements.
```

However, if the global placement result cannot be legalized, your program score will be zero.

```
[Error] Fail to Legalize! The global placement result cannot be
legalized.
```

The following information is the baseline wirelength of each testcase. For each testcase, if your wirelength is greater than the baseline wirelength, you will get 0 points on this testcase.

Testcase	Wirelength
public1	290047502
public2	26105833
public3	2392482780

Notes:

- Please use the following command to compile your program.
  - Go into directory "`src/`", enter "`make`" to compile your program and generate the executable file, called "`hw4`", which will be in directory "`bin/`".
  - Go into directory "`src/`", enter "`make clean`" to delete your executable file.
- Please use the following command line to execute the program:

  ```
  $ ./hw4 <.aux file> <.gp.pl file>
  ```

  For example:

  ```
  $ ./hw4 ../testcases/public1/public1.aux ../output/public1.gp.pl
  ```
- If you implement parallelization, please name the executable file of your parallel version as "hw4_parallel" and name the executable file of your sequential version as "hw4".
- Use arguments to read the file path. Do not write the file path in your code.
- 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 HW4_grading.sh. If we cannot compile or execute your program by the script, you will get 0 points on your programming score.
- 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.)