CS6135 VLSI Physical Design Automation

Homework 4: Global Placement

Due: 23:59, December 31, 2020

1. Problem Statement

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).

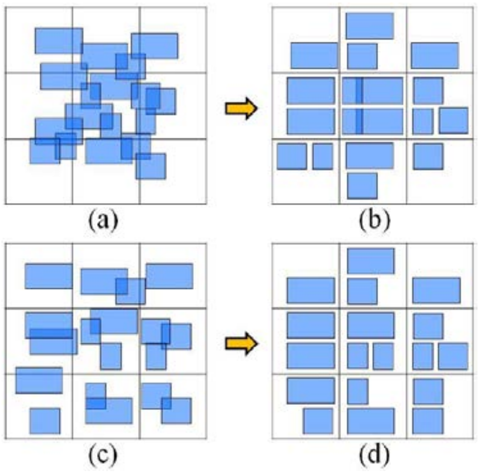


Figure 1. The process of global placement and legalization

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 */
    string name();
    double x(); // coordinate of the bottom-left corner
    double y();
    double centerX();
    double centerY();
    double width();
    double height();
    double area();
    unsigned numPins();
    Pin& pin(unsigned index); // index: 0 ~ numPins()-1
    /* set functions */
    // use this function to set the module 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:
    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
    // output global placement result file for later stages
    outputBookshelfFormat(string fileName);
};
```

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 coordinates of modules (to move blocks). 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. Compile & Execution

To compile the program, simply type:

```
$ make
```

Please use the following command line to execute the program:

```
$ ./hw4 -aux <aux file>
```

For example:

```
$ ./hw4 -aux testcases/ibm01/ibm01-cu85.aux
```

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) What tricks did you do to speed up your program or to enhance your solution quality?

(5) Please compare your results with the previous top 5 students' results and show your advantage either in runtime or 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 program is too slow, then what could be the bottleneck of your program? 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			**Runtime(s)**		
Ranks	ibm01	ibm05	ibm09	ibm01	ibm05	ibm09
1	86731510	11640852	537572709	180	380	894
2	76524759	10287864	720387019	505	481	1176
3	212729459	9962963	1640571405	1	4	11
4	123499363	12490103	1558917733	59	143	322
5	315825188	28470167	2644255425	8	40	56

6. Required Items

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

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

(2) `hw4`: an executable file which is generated by `Makefile`.

(3) `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_109062501.tar.gz HW4/
```

7. Evaluation

This programming assignment will be graded based on the (1) correctness of the program, (2) solution quality, (3) runtime and (4) report. Please check these items before your submission.

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: ibm01-cu85


Global HPWL: 735305578   Time:    0.0 sec (0.0 min)
 Legal HPWL: 688045191   Time:    1.0 sec (0.0 min)
Detail HPWL: 318658248   Time:    7.0 sec (0.1 min)
 ===================================================================
       HPWL: 318658248   Time:    8.0 sec (0.1 min)
```

However, if the global placement result cannot be legalized, the solution quality will be judged by the following cost function:

$$W \times (1 + \text{"}\boldsymbol{Scaled\ Overflow\ per\ bin}\text{"})$$

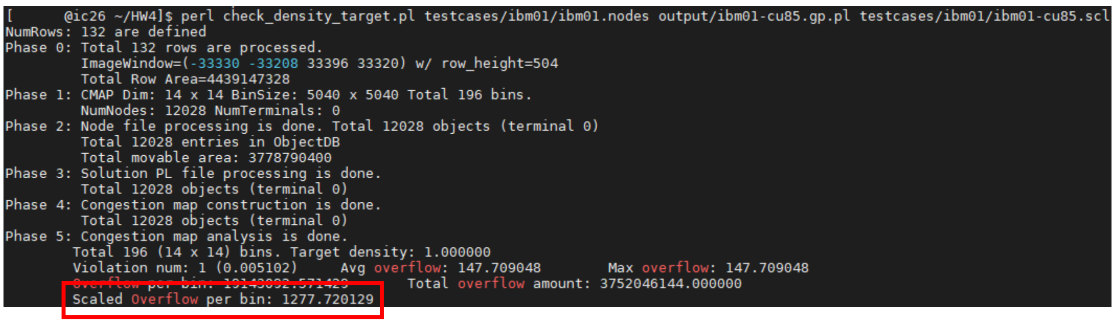
where W is the total wirelength derived from the global placement stage. That is, if you result cannot be legalized by the legalizer, the resultant HPWL will be penalized and become extremely large. The "$\boldsymbol{Scaled\ Overflow\ per\ bin}$" can be found by using the following script:

```
$ perl check_density_target.pl <nodes file> <gp.pl file> <scl file>
```

For example:

```
$ perl check_density_target.pl testcases/ibm01/ibm01.nodes output/ibm01-cu85.gp.pl testcases/ibm01/ibm01-cu85.scl
```

Then "$\boldsymbol{Scaled\ Overflow\ per\ bin}$" can be checked on the screen as follows:



```
[      @ic26 ~/HW4]$ perl check_density_target.pl testcases/ibm01/ibm01.nodes output/ibm01-cu85.gp.pl testcases/ibm01/ibm01-cu85.scl
NumRows: 132 are defined
Phase 0: Total 132 rows are processed.
         ImageWindow=(-33330 -33208 33396 33320) w/ row_height=504
         Total Row Area=4439147328
Phase 1: CMAP Dim: 14 x 14 BinSize: 5040 x 5040 Total 196 bins.
         NumNodes: 12028 NumTerminals: 0
Phase 2: Node file processing is done. Total 12028 objects (terminal 0)
         Total 12028 entries in ObjectDB
         Total movable area: 3778790400
Phase 3: Solution PL file processing is done.
         Total 12028 objects (terminal 0)
Phase 4: Congestion map construction is done.
         Total 12028 objects (terminal 0)
Phase 5: Congestion map analysis is done.
        Total 196 (14 x 14) bins. Target density: 1.000000
        Violation num: 1 (0.005102)     Avg overflow: 147.709048        Max overflow: 147.709048
        Overflow per bin: 19143092.571429       Total overflow amount: 3752046144.000000
        Scaled Overflow per bin: 1277.720129
```

Note that solutions which can be legalized will get better scores than those that cannot be legalized.

8. Grading

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

Notes:

- Please do not modify the output message.
- Program must be terminated within 20 minutes for each testcase.
- Grading is based on the total wirelength (primary) and runtime (secondary).