CS6135 VLSI Physical Design Automation

Homework 4: Placement Legalization

Due: 23:59, December 25 2022

1. Introduction

In this homework, you are asked to adapt and implement an existing algorithm, published in the ISPD-08 paper entitled "**Abacus: fast legalization of standard cell circuits with minimal movement**" by Spindler, Schlichtmann and Johannes, to legalize a given global placement result with minimal total displacement (measured by Euclidean distance).

2. Problem Description

(1) Input:

- A set of standard cells (and blockages), where each standard cell (or blockage) has a rectangular shape specified by its width and height. The design is composed of single-row height movable cells, and multiple-row height fixed blockages.
- The original coordinates of each cell (or blockage).
- Chip specification, such as the coordinates of each row, the uniform row height, the width of sites, etc.

(2) Output:

The coordinates of each cell (or blockage) after legalization are finished. The coordinates of each cell or blockage are specified by its lower-left corner.

(3) Objective:

Cells are not allowed to be rotated, but they are only allowed to be moved instead. The total displacement of the legalization result and the runtime of you program should be as small as possible subject to the following constraints.

1. Aligning constraint: Each cell is not allowed to cross multiple rows, and must align its left boundary with a site of the corresponding row where the cell locates. If this constraint is violated, your grade will be set to 0.
2. Non-overlapping constraint: No two cells overlap with each other. If this constraint is violated, your grade will be set to 0.

3. Input File

(1) The *.aux* file:

The .aux file specifies the filenames which contain the design information you should know. Here is an example:

```
RowBasedPlacement      : adaptec1.node adaptec1.pl adaptec1.scl
// RowBasedPlacement   : .node file .pl file .scl file
MaxDisplacement        : 120
// MaxDisplacement     : maximum displacement threshold
```

- *MaxDisplacement* is a maximum displacement threshold for each cell.

(2) The *node* file:

The .node file specifies the name, width, height and the other information about each cell/blockage in the global placement result. Each line specifies a single cell/blockage. Here is an example:

```
NumNodes          : 210967
// NumNodes       : number of cells and blockages
NumTerminals      : 63
// NumTerminals   : number of blockages


o0          8         12
o0211446  2304        24    terminal
// nodeName   width   heigh   moveType
      ⋮
```

- *nodeName* is an arbitrary-length alpha-numeric string, and is case-sensitive.
- *width* is a multiple of the site width.
- *height* is a multiple of the row height.
- *moveType* is an optional string. If not given, it means a movable cell; *terminal* means a fixed blockage.

(3) The *.pl* file:

The .pl file specifies the original coordinates of each cell/blockage in the global placement result. Here is an example:

```
o0         6198.94   4058.75   : N
o211446  627         11103     : N /FIXED
// nodeName x-coordinate y-coordinate : orientation moveType
        ⋮
```

- *Orientation* is always N.
- *moveType* is an optional string. If not given, it means a movable cell; /FIXED means a fixed blockage.

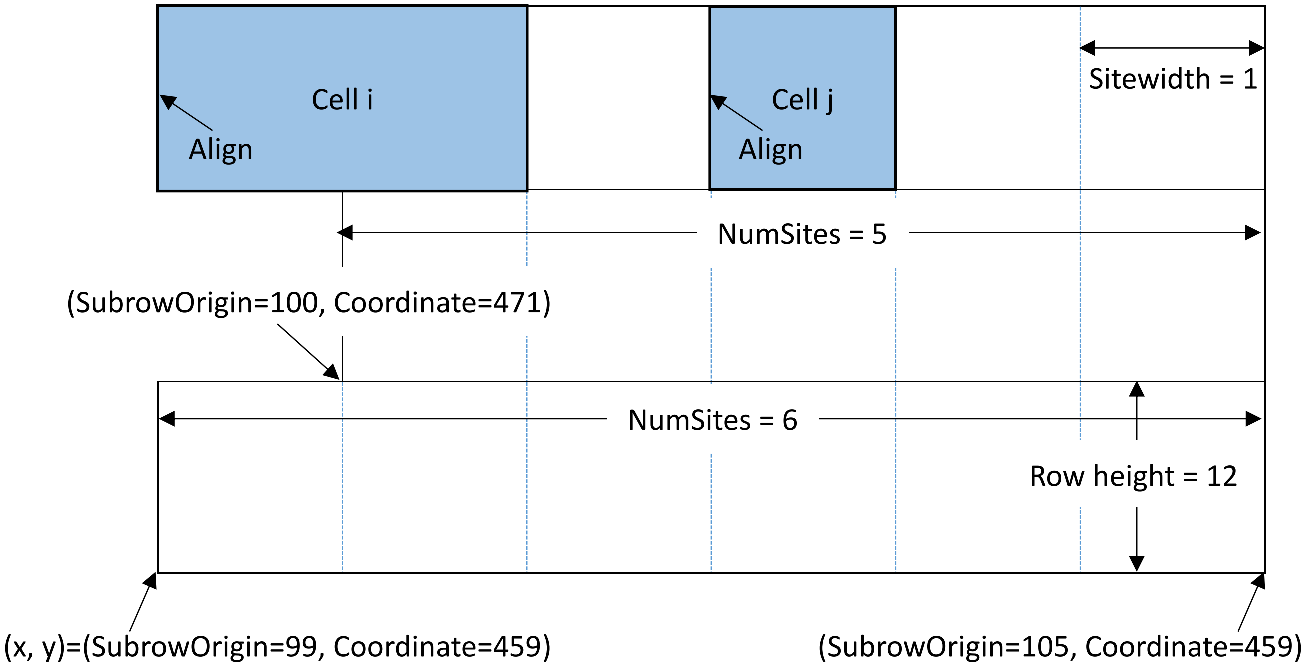
(4) The *.scl* file:

The .scl file specifies the information of rows in the placement region. Here is an example:

```
NumRows : 890
//NumRows : number of rows

CoreRow Horizontal //1st row
     Coordinate    : 459
   //Coordinate    : y-coordinate of row
     Height        : 12
   //Height        : row height
     Sitewidth     : 1
   //Sitewidth     : width of a site
     NumSites      : 10692
   //NumSites      : number of sites
     SubrowOrigin : 459
   //SubrowOrigin : x-coordinate of row
End
        ⋮
```

- *Coordinate* represents the y-coordinate of the bottom boundary of row.
- *Sitewidth* is an identical integer that represents the width of a site.
- *SubrowOrigin* represents the x-coordinate of the left boundary of the row.


Cell i
Align
Cell j
Align
Sitewidth = 1
NumSites = 5
(SubrowOrigin=100, Coordinate=471)
NumSites = 6
Row height = 12
(x, y)=(SubrowOrigin=99, Coordinate=459)
(SubrowOrigin=105, Coordinate=459)

4. Output File

(1) The ***.result*** file:

The .result file specifies the legalization result containing the coordinates of each cell (or blockage). The filename is the same as the .aux file. Here is an example:

```
o0   6295   4119
// nodeName   x-coordinate   y-coordinate
o1   3064   8619
        ⋮
```

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.

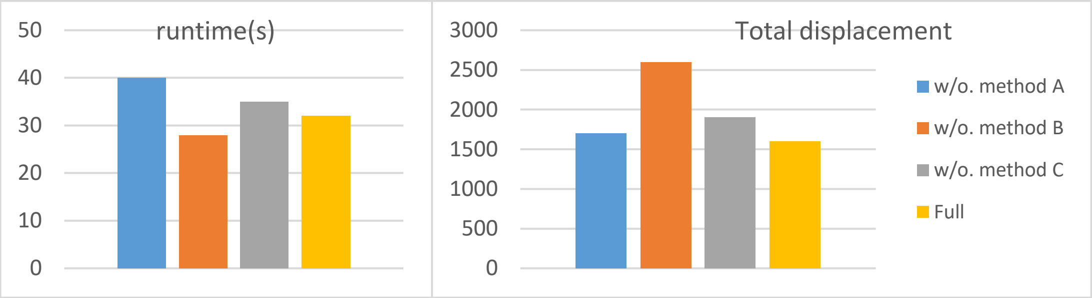
(1) Your name and student ID

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

(3) The total displacement, the maximum displacement and the runtime of each testcase. Notice that the runtime contains I/O, constructing data structures, computing parts, etc. (You need to provide a screenshot for the result of HW4_grading.sh.)

(4) The details of your implementation. If there is anything different between your implementation and the algorithm in the ISPD-08 paper, please reveal the difference(s) and explain the reasons.

(5) What tricks did you do to speed up your program or to enhance your solution quality? Also plot the effects of those different settings like the ones shown below.



(6) 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?

Top 5 students' results

	Total Disp. (10^6)					Runtime (sec)				
Ranks	**ibm01**	**ibm07**	**ibm09**	**ada1**	**ada3**	**ibm01**	**ibm07**	**ibm09**	**ada1**	**ada3**
1	5.5	27.86	41.96	3.07	5.08	0.06	0.23	0.35	2.35	4.17
2	6.1	30.24	47.92	3.42	5.47	0.07	0.27	0.33	1.74	3.44
3	6.24	31.63	52.21	3.32	5.21	0.05	0.17	0.2	1.07	3.26
4	6.04	31.11	50.28	3.38	5.27	0.28	1.22	1.22	12.81	58.14
5	5.57	29.9	46.5	3.69	5.48	0.1	0.42	0.48	28.21	99.99

(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 `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, `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 the TA to verify.

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

(4) `CS6135_HW4_${STUDENT_ID}_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_11062500.tar.gz HW4/
```

8. Grading

- ✓ **81%:** Total displacement and maximum displacement of each testcase; hidden testcases included. There are 9 cases in total, 9 points for each case, and the following rules will be used to grade them:
  - ● 5 points for legal placement result.
  - ● 2 points for total displacement.
  - ● 2 points for satisfying the maximum displacement constraint.
- ✓ **19%:** The completeness of your report.
- ✓ **5% Bonus**: Parallelization. Please specify your system specification.

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 "`hw4`", 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.

  ```
  $ ./hw4 *.aux *.result
  ```

  E.g.:

  ```
  $ ./hw4 ../testcase/adaptec1/adaptec1.aux ../output/adaptec1.result
  ```

- If you implement parallelization, please name your parallel-version executable file as "`hw4_parallel`" and name your sequential-version executable file as "`hw4`".
- Use arguments to read the file path. Do not write the file path in your code.
- Your program must be terminated within 3 minutes for each testcase.

- We will test your program by shell script with GCC 9.3.0 on ic51. Please make sure your program can be executed by **HW4_grading.sh** (Your programming score will be 0 if your program does not pass the shell script).
- Note that any form of plagiarism is strictly prohibited. If you have any question about the homework, please contact TAs. (If you have any questions about C/C++ programming, please google it by yourself.)