CS6135 VLSI Physical Design Automation

Homework 3: Detailed Placement

Due: 23:59, November 18, 2025

1. Introduction

In this assignment, you are asked to implement a detailed placer to minimize the total wirelength of a given placement. Given a legalized placement (including both movable modules and fixed modules) and the corresponding netlist, the detailed placer must keep the positions of all fixed modules unchanged while adjusting the positions of movable modules to reduce the overall wirelength.

2. Problem Description

(1) Input:

- ✓ Library Exchange Format (LEF) file: site and module definition.
- ✓ Design Exchange Format (DEF) file: row definition, module coordinate and orientation, I/O pad definition and coordinate, and netlist.

(2) Output:

- ✓ The final placement result (DEF file)

(3) Objective:

Minimize the total wirelength (in database unit, refer to Section 3.2) of the placement by adjusting movable modules. The total wirelength W of a set N of nets (listed in NETS section in DEF file) is computed as:

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

where n_i denotes a net in N, $HPWL(n_i)$ represents the half-perimeter wirelength of net n_i, and special nets (listed in SPECIALNETS section in DEF file) are excluded from the calculation. To simplify wirelength calculation, each pin of a module m_i is located at the bottom-left coordinates of m_i. After performing detailed placement, the resulting placement must remain legalized, meaning: 1) No modules overlap; 2) The lower-left corner of each movable module must align exactly with its corresponding row and site.

3. Input File

(1) The LEF file:

The LEF file specifies the library information. Here is an example:

```
  …
# Define unit conversion between LEF coordinates and microns
UNITS
   DATABASE MICRONS 2000 ;          # 1 micron = 2000 database units
END UNITS

# Define a site type used for standard cells
SITE CoreSite
   CLASS CORE ;                     # This site is for core (standard cell) placement
   SIZE 0.1 BY 1.2 ;                # Site width = 0.1 μm, height = 1.2 μm
END CoreSite

# Define a standard cell (logic cell)
MACRO XOR2X1
      CLASS CORE ;                  # This is a core cell (not a block or pad)
      FOREIGN XOR2X1 0 0 ;          # Name and reference origin for external tools
      ORIGIN 0 0 ;                  # Local origin of the cell (lower-left corner)
      SIZE 1.400000 BY 1.200000 ;   # Cell width = 1.4 μm, height = 1.2 μm
      SYMMETRY X Y ;                # Cell can be mirrored along X or Y axis
      SITE CoreSite ;               # This cell uses the CoreSite definition
      ...
END XOR2X1
  …
# Define a block (macro cell)
MACRO MEM1
   CLASS BLOCK ;                    # This is a large block (e.g., memory, hard macro)
   ORIGIN 0 0 ;
   FOREIGN MEM1 0 0 ;
   SIZE 426.965 BY 114.215 ;
   SYMMETRY X Y R90 ;
   ...
END MEM1
```

(2) The DEF file:

The DEF file specifies the design information. Here is an example:

```
# Define the design name
DESIGN public1 ;
UNITS DISTANCE MICRONS 2000 ;
DIEAREA ( 0 0 ) ( 1745600 1178400 ) ;
# Define standard-cell placement rows
ROW CORE_ROW_0 CoreSite 6000 6000 FS DO 8668 BY 1 STEP 200 0 ;
# “CoreSite” is the site name defined in LEF
# (6000 6000) = row origin, FS = flipped and shifted (orientation)
# DO 8668 BY 1 = number of sites horizontally and vertically
# STEP 200 0 = site-to-site distance in x and y directions
  …
# Define all placed components (standard cells or macros)
COMPONENTS 72094 ;
   - inst0 SDFFHQX2
      + PLACED ( 1378200 627600 ) N ;    # Movable module name, cell type, position, and orientation
   - inst13083 MEM2
      + FIXED ( 881872 326792 ) N ;      # Fixed module name, cell type, position, and orientation
...
END COMPONENTS
# Define I/O pins on the chip boundary
PINS 1211 ;
- pin999 + NET pin1 + DIRECTION OUTPUT + USE SIGNAL   # Pin name, connected net name, …
   + LAYER Metal7 ( -100 0 ) ( 100 1040 )             # Metal layer and pin shape
   + PLACED ( 1292800 0 ) N ;                         # Pin location and orientation
   ...
END PINS
# Define electrical connections between modules and pins
NETS 72410 ;
- net2110
   ( PIN pin998 ) ( inst46290 Q ) ( inst11143 Q ) ( inst2565 SI ) ( inst1817 SI )
   ;   # Each net lists all connected pins or module terminals
   ...
END NETS
END DESIGN
```

All dimensions in LEF files are expressed in microns, whereas all coordinates in DEF files are expressed in database units. Please remember to perform the proper unit conversion when interpreting or comparing the two files.

4. Output File

(1) The DEF file:

After performing detailed placement, output the module coordinates in a DEF file. To simplify the output format, only the coordinates and orientations of modules in the COMPONENTS section must be updated. Each module's orientation must match the orientation of the row on which it is placed. All other parts of the output DEF must remain exactly the same as those in the input DEF.

5. Language/Platform

(1) Language: C/C++

(2) Platform: Unix/Linux

6. Report

Your report should contain the following content, and you can add more as you wish.

(1) Your name and student ID

(2) The final wirelength and the runtime of each testcase. Paste the screenshot of the result of running the **HW3_grading.sh**.

(3) In addition to the formats introduced above, both LEF and DEF files contain many other types of information. Please list three additional elements from LEF and three from DEF, respectively, and describe the meaning of each.

(4) Describe the details of your implementation. You may use flowcharts and/or pseudocode to illustrate your algorithm. Explain which detailed placement methods you adopted (e.g., local reordering, global swap, independent set matching) and provide a clear description of how each method was implemented.

(5) 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_${StudentID}_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_114000000.tar.gz HW3/
```

8. Grading

✓ 50%: Outperform the baseline in public cases to get full marks for each case. The wirelength (in database unit) of baseline for each public case is listed below.

Testcase	HPWL	Testcase	HPWL
public1	83,036,000	public3	31,265,938,140
public2	3,625,878,314	public4	41,714,599,410

✓ 30%: Solution quality. The solution quality is evaluated based on the wirelength for each hidden case. Assume there are three hidden cases in total. For each hidden case, the score is computed using the following formula:

Hidden Case Score = 5 + 5 × max((avg(x) − min(x)) / (avg(x) − self(x)), −1)

where:
- ➢ **self(x)** is the wirelength of your solution,
- ➢ **avg(x)** is the average wirelength of all submissions, and
- ➢ **min(x)** is the minimum wirelength among all submissions.

✓ 20%: The completeness of your report

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 <input LEF> <input DEF> <output DEF>
```

  E.g.:

```
$ ./hw3 ../testcase/public1.lef ../testcase/public1.def ../output/public1.def
```

- Use arguments to read the file path. Do not write the file path in your code.
- Program must be terminated within 300 seconds for each testcase.
- Please use ic21, ic22, or ic51 to test your program.
- We will test your program by a shell script with GCC 9.3.0 on ic21. 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.
- Note that any form of plagiarism is strictly prohibited, including the code found on GitHub and the code from any student who took this course before. If you have any problem, please contact TA.
- Notes on AI Tool Usage
  (1) Do not submit code generated directly by AI or rewritten from public repositories.
  (2) You may use AI for debugging or code improvement, but you must disclose how it was used.
  (3) Clearly state in comments or a separate note which AI tools you used and for what purpose.
  (4) You are responsible for verifying correctness and being able to explain your final code.

References:

[1] K. Doll, F. M. Johannes and K. J. Antreich, "Iterative Placement Improvement by Network Flow Methods," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 10, pp. 1189-1200, 1994.

[2] A. B. Kahng, P. Tucker and A. Zelikovsky, "Optimization of Linear Placements for Wirelength Minimization with Free Sites," in *Proceedings of Asia and South Pacific Design Automation Conference*, 1999.

[3] U. Brenner, and J. Vygen, "Faster Optimal Single-Row Placement with Fixed Ordering," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, 2000.

[4] M. Pan, N. Viswanathan and C. Chu, "An Efficient and Effective Detailed Placement Algorithm," in *Proceedings of International Conference on Computer-Aided Design*, 2005.

[5] Y. Lin, W. Li, J. Gu, H. Ren, B. Khailany and D. Z. Pan, "ABCDPlace: Accelerated Batch-Based Concurrent Detailed Placement on Multithreaded CPUs and GPUs," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 5083-5096, 2020.