CS6135 VLSI Physical Design Automation

Homework 4: Analog Device Placement Considering Symmetry Constraints

Due: 23:59, May 15, 2025

1. Introduction

In this homework, you are asked to implement an analog device placer considering symmetry constraints, using the benchmark circuits published in the 2009 IEEE TCAD paper entitled "Analog Placement Based on Symmetry-Island Formulation" by Lin *et al*.

2. Problem Description

(1) Input:

- A set B of rectangular hard blocks corresponding to different analog devices, where each block b_i in B has its width and height.
- A set of symmetry constraints containing one or multiple symmetry groups. Each symmetry group consists of more than one symmetry pairs, (b_i, b_i'), and/or self-symmetric device.

(2) Output:

- The coordinates $(x_i,\ y_i)$ of the lower-left corner of each block b_i, as well as the rotation status (1 for rotated, and 0 for unrotated).

(3) Objective:

By assuming each block can be rotated by 90 degrees, the objective is to minimize the total area of the analog device placement result. The coordinate of each block in the placement result should be positive. Total area is calculated by the minimum rectangle bounding all the blocks with the lower-left corner starting from (0, 0). Furthermore, the placement result is subject to the following constraints.

1. Symmetry constraint: Each symmetry group must form a symmetric placement with respect to a vertical/horizontal symmetry axis.
2. Non-overlapping constraint: No two blocks overlap with each other.

3. Input File

(1) The *.txt* file:

The .txt file specifies the information about the analog devices.

```
NumHardBlocks 10
// NumHardBlocks number of hard blocks
HardBlock hb0 12 331
// HardBlock block name width height
        ⋮

NumSymGroups 2
// NumSymGroups number of symmetry groups
SymGroup sg0 2
// SymGroup group name number of pair- and self-symmetry blocks
SymPair sp0 sp1
// SymPair block name1 block name2
SymSelf ss0
// SymSelf block name
SymGroup sg1 3
SymPair sp2 sp3
SymPair sp4 sp5
        ⋮
```

4. Output File

(1) The *.out* file:

The .out file specifies the analog device placement result including the total area of the bounded rectangle, the total number of the analog devices, and the coordinates of the lower-left corner of each analog device block with/without rotation.

```
Area 11324900

NumHardBlocks 49
hb0 0 0 1
// block name lower-left corner coordinates (x,y) rotated
hb1 0 1708 0
// block name lower-left corner coordinates (x,y) unrotated
        ⋮
```

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 area and the runtime of each testcase, respectively. Paste the screenshot of the result of running the **HW4_grading.sh**.

(4) The details of your implementation.

(5) Please describe your method of your initial placement.

(6) What tricks did you do to speed up your program or to enhance your solution quality? Please use **HW4_printer** to generate the image to compare different stages (e.g., initial placement → trick 1 → trick 2 → final result) of your placement results.

(7) 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, 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_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_113000000.tar.gz HW4/
```

8. Grading

✓ 50%: Outperform the baseline in public testcases. The area of baseline for each public testcase is listed below. For hidden testcases, you only need to generate a valid result.

Case name	public1	public2	public3
Area	52,660,568	695,913	636,576

✓ 30%: The area of each testcase, hidden testcases included.

✓ 20%: The completeness of your report

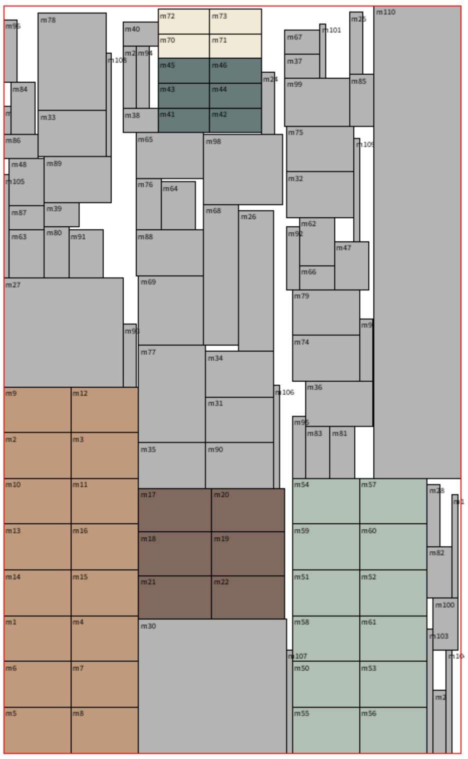
Notes:

- Make sure the following commands can be executed.
  - Go into directory "`src/`", enter "`make`" to compile your program and generate the executable file, called "`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 *.txt *.out
  ```

  E.g.:

  ```
  $ ./hw4 ../testcase/public1.txt ../output/public1.out
  ```
- Use arguments to read the file path. Do not write the file path in your code.
- Program must be terminated within 5 minutes for each testcase.
- Please use ic21, ic22 to test your program.
- We will test your program by a shell script with GCC 9.3.0 on the servers mentioned above. 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.
- For each testcase, you should use **HW4_printer** to draw your result like the following figure and paste the figure on your report.



- 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.