Andy, Yu-Guang Chen

EE6094 CAD for VLSI Design
Programming Assignment 3: Analog Placement

(Due: 23:59:59, 2024/05/26)

- Version 1: 2024.04.17 22:40:00
- Version 2: 2024.04.25 22:00:00
  - Update scripts usages in **Workflow**
  - Add constraints in **Correctness**
  - Adjust the cost function in **Quality**
  - Update available time in **Contact**
- Version 3: 2024.05.04 23:00:00
  - Extend due date
  - Update available time in **Contact**
  - Add the number of test cases in **Grading**
  - Add reference [11] in **Reference**
  - Add explanations about <col_multiple> and <row_multiple> in **Example**

Introduction

In the context of creating analog device-level layouts, an automated placement tool needs to go beyond basic rectangle packing. It should possess analog-specific capabilities to address critical requirements [1] including:

- **Symmetry constraint**
  The main reason of symmetric placement and routing is to match the layout-induced parasitic capacitances in the two halves of a group of devices.
- **Device merging constraint**
  Device mismatches are due both to random events in the manufacturing process. In order to reduce the degree of electrical mismatch due to area effects, matching groups of cells can be defined, all members being constrained to the same orientation and device variant.
- **Module library generator** [2][11]
  A built-in library of predefined modules allows users to reshape layouts during the placement process.

These three requirements can be referenced in the subsequent paragraphs, which describe files with the extensions ***.sym***, ***.group*** and ***.block***.

Workflow

You are asked to implement your own analog placer. We covered various analog placement algorithms in class, including B*-tree [3][6][7] and Sequence Pair [1][4][5]. Choose a feasible algorithm to complete your programming assignment. Refer to Figure 1, the entire workflow is as follows:

1. Set up the work environment
   - Command line

   ```
   source /home/CAD112/PA3/env.cshrc
   ```

2. Run ***BuildingBlockGenerator*** to generate building blocks that meet process requirements and are feasible. The script will generate a GDSII File in the directory where <building_block_file> is located for visualizing the results.
   - Command line

   ```
   BuildingBlockGenerator <netlist_file> <tech_file> <device_group_file>
   <building_block_file>
   ```

3. Execute your analog placer to generate the placement results. You have to compile your source code to the executable file in the name of your student ID.
   - Command line

   ```
   ./<student_id> <expected_aspect_ratio> <netlist_file>
   <symmetry_constraint_file> <building_block_file> <output_file>
   ```

4. Run ***Verifier*** to validate your results.
   - Command line

   ```
   Verifier <expected_aspect_ratio> <netlist_file> <device_group_file>
   <symmetry_constraint_file> <building_block_file> <output_file>
   ```

5. Utilize ***GdsGenerator*** to convert the placement results into a GDSII File for visualization purposes, if desired. The script will generate a GDSII File in the directory where <output_file> is located for visualizing the results.
   - Command line

   ```
   GdsGenerator <netlist_file> <tech_file> <output_file>
   ```

Note

- GDSII (Graphical Design System/Graphic Data System) is a data conversion format for integrated circuit layouts. It primarily serves as input data for EDA Tools, enabling the presentation of circuit layout designs in 2D across various layers.
- You can download KLayout [8], an open-source layout editor, to assist you in visualizing your placements by opening the converted GDSII files.

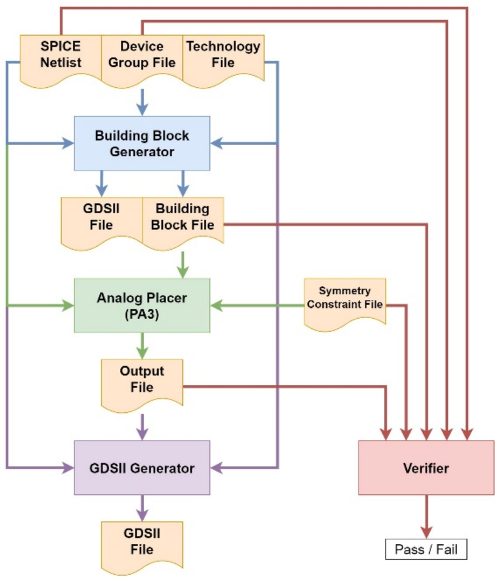


Figure 1: Workflow chart

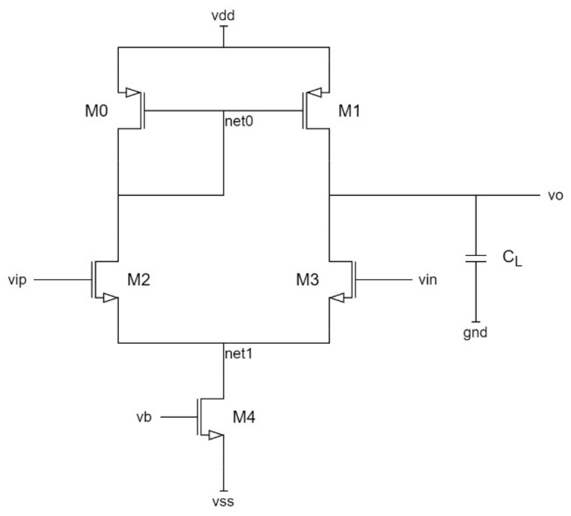


Figure 2: Circuit schematic

Example (Figure 2)

Input files

- SPICE [9] netlist file (.netlist)

```
.SUBCKT EXAMPLE Vb vin vip vo gnd! vdd! vss!
MM0 net0 net0 vdd! vdd! PMOS W=3.5u L=0.5u m=2
MM1 vo net0 vdd! vdd! PMOS W=3.5u L=0.5u m=2
MM2 net0 vip net1 vss! NMOS W=4u L=0.5u m=1
MM3 vo vin net1 vss! NMOS W=4u L=0.5u m=1
MM4 net1 vb vss! vss! NMOS W=4.8u L=0.5u m=4
CL vo gnd! 1p
.ENDS
```

- Multiplier (m)
  In advanced technology nodes, multiple transistor instances are used to optimize circuit performance. For more information, please refer to [11].

- Device group file (.group)

```
BB0 MM0 MM1 4 1
```

- Technology file (.tf)

```
deviceRule {
        minPoly2GateExtension = 0.23
        minCo2GateSpace = 0.15
        minCoWidth = 0.23
        minCoSpace = 0.25
        minCo2OdEnclosure = 0.12
        minOdSpace = 0.4
        minPolySpace = 0.28
        minOd2ImpEnclosure = 0.12
        minCo2MetEnclosure = 0.1
}
layerTable {
        Diffusion = 1
        Poly = 25
        Contact = 40
        Pplus = 10
        Nplus = 11
        Metal = 45
}
```

- Symmetry constraint file (.sym) [7]

```
Symmetry0 BB0
Symmetry0 MM2 MM3
Symmetry0 MM4
```

- Format
  1. Self-symmetric module
     <symmetry_group_name> <building_block_name>
  2. Symmetry pair
     <symmetry_group_name> <building_block_name_1>
     <building_block_name_2>

Intermediate files

- Building block file (.block)

```
BB0 MM0 MM1 (4.99 4.24 4 1)
MM2 (1.9 4.74 1 1)
MM3 (1.9 4.74 1 1)
MM4 (1.9 22.16 1 4) (2.93 11.08 2 2) (4.99 5.54 4 1)
```

- Format
  (1) <building_block_name> <all_device_names> (<width> <height> <col_multiple> <row_multiple>) …
  (2) <device_name> (<width> <height> <col_multiple> <row_multiple>) …

- Note
  <col_multiple> and <row_multiple> represent the number of parallel instances in two dimensions, determined by the parameter multiplier.

- GDSII file is used to visualize building blocks (.block.gds)

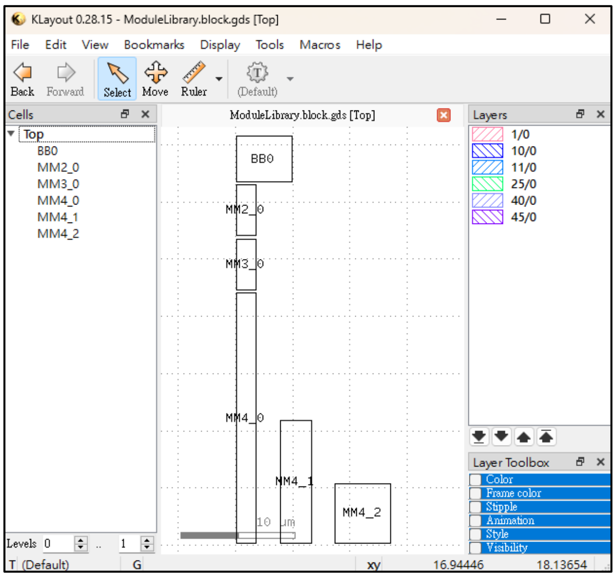


Figure 3: Building blocks in Klayout

Output files

➢ Placement result file (.output)

```
17.92
72.4548
4.99 14.52
BB0 MM0 MM1 0.0 10.28 (4.99 4.24 4 1)
MM2 0.595 5.54 (1.9 4.74 1 1)
MM3 2.495 5.54 (1.9 4.74 1 1)
MM4 0.0 0.0 (4.99 5.54 4 1)
```

■ Format

(1) Line 1: Total HPWL
(2) Line 2: Chip area
(3) Line 3: Chip width and chip height
(4) Line 4 to end:
   i. <building_block_name> <all_device_names> <x> <y> (<width> <height> <col_multiple> <row_multiple>)
   ii. <device_name> <x> <y> (<width> <height> <col_multiple> <row_multiple>)

➢ GDSII file is used to visualize placement result (.output.gds)

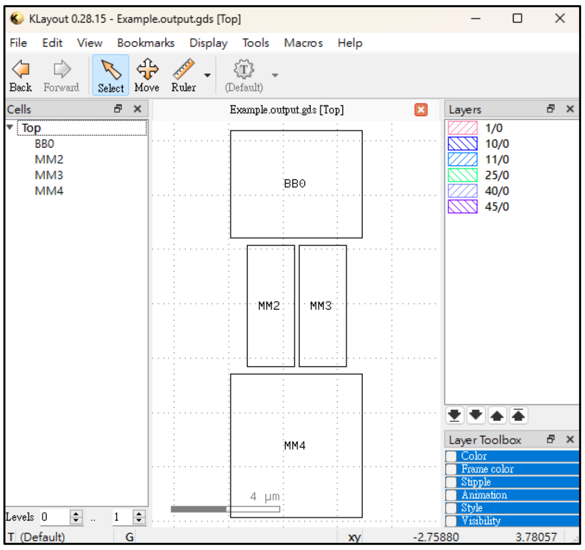

KLayout 0.28.15 - Example.output.gds [Top]
File Edit View Bookmarks Display Tools Macros Help
Back Forward Select Move Ruler (Default)
Cells
Top
BB0
MM2
MM3
MM4
Example.output.gds [Top]
BB0
MM2
MM3
MM4
4 μm
Layers
1/0
10/0
11/0
25/0
40/0
45/0
Layer Toolbox
Color
Frame color
Stipple
Animation
Style
Visibility
Levels 0 .. 1
T (Default) G xy -2.75880 3.78057

Figure 4: Placement result in Klayout

Correctness

We will use ***Verifier*** to check the correctness of your placement results. Each test case will be scored independently. Your output must meet the following conditions:

(1) Contain all devices

(2) With correct HPWL, area, width, height and building block pattern

(3) No overlap between any devices

Quality

In the analog placement problems, violations of symmetry constraints are unacceptable. However, for the sake of difficulty, we will consider such violations as part of the quality of results in this programming assignment. We divide the grading rules into two parts and grade them separately. Note that the quality points will be 0 if your result does not meet all correctness constraints.

Part 1 (Total 20%)

➢ **Meet symmetry constraints [7]**

In this work, we only consider placement symmetrical about the vertical axis.

- Self-symmetric module

$$CenterX_{building\ block} = X_{symmetry\ axis}$$

- Symmetry pair

$$\frac{CenterX_{building\ block\ 1} + CenterX_{building\ block\ 2}}{2} = X_{symmetry\ axis}$$

$$MaxY_{building\ block\ 1} = MaxY_{building\ block\ 2}$$

$$MinY_{building\ block\ 1} = MinY_{building\ block\ 2}$$

Part 2 (Total 20%)

We select three of the popular design metrics, and form the cost function as below. We will judge the quality of the results through the cost function. The smaller the better.

$$cost = (0.25 \times ChipArea + 0.75 \times TotalHPWL) \times (1 + (\frac{AR - ExpectedAR}{3})^2)$$

- **Minimize chip area**

  Minimize the bounding box area that frames all devices.

- **Minimize half-perimeter wire length (HPWL)**

  The HPWL represents half of the perimeter of the bounding box that encloses the centers of all the building blocks within a network.

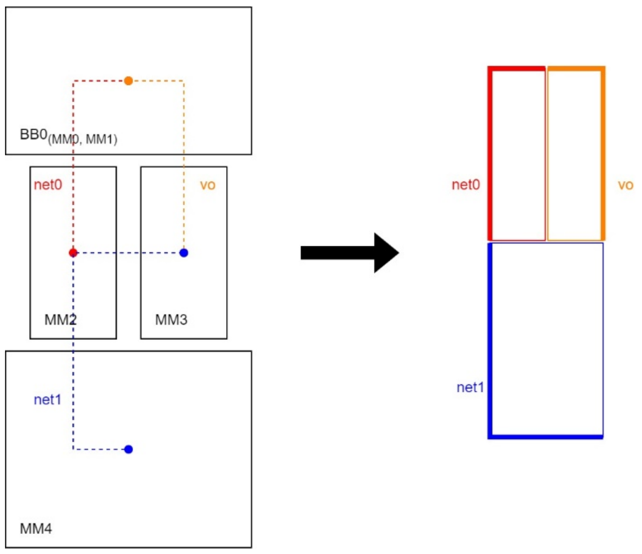


Figure 5: Take Figure 2 and Figure 4 as an example to illustrate how to calculate HPWL. Since MM0 and MM1 are merged into BB0, all connections related to MM0 and MM1 will be calculated by the center of BB0, and the unmerged building blocks will be calculated by their own center.

- **Aspect ratio (AR)**

  Since the slender chip layout is not easy to integrate with other modules, the aspect ratio of the chip will be limited to a certain ratio. Aspect ratio defined as below:

$$AspectRatio = \max\left(\frac{ChipWidth}{ChipHeight}, \frac{ChipHeight}{ChipWidth}\right)$$

Requirement

1. You must write this program in C or C++. No open-source codes are allowed to use. (i.e., you MUST implement the tool by yourself). The run time of your program is limited to at most 10 minutes per testcase. You can use optimization flags [10] at compile time to speed up your program. We will verify your program on workstation. In the other words, you have to make sure your program can be compiled successfully and executed correctly on

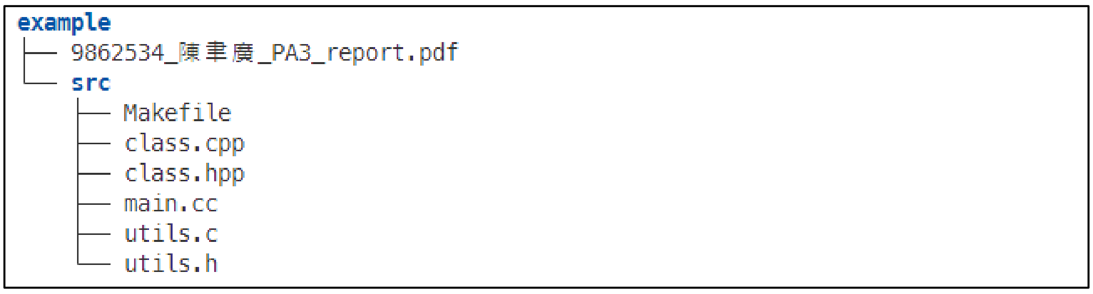
workstation. The workstation information is shown below:

- Operating System: CentOS Linux 7 (Core)
- Kernel: Linux 3.10.0-229.11.1.el7.x86_64
- Architecture: x86-64
- Compiler: gcc version 13.2.0 (GCC)

2. The files that need to be submitted to ee-class are as follows:

   (1) A folder named "src", contains all source code (.h, .cpp, etc.) and a Makefile. Except for Makefile and executable file, there are no restrictions on the file names in this folder.

   (2) A report named <student_id>_<name>_PA3_report.pdf



```
example
├── 9862534_陳聿廣_PA3_report.pdf
└── src
    ├── Makefile
    ├── class.cpp
    ├── class.hpp
    ├── main.cc
    ├── utils.c
    └── utils.h
```

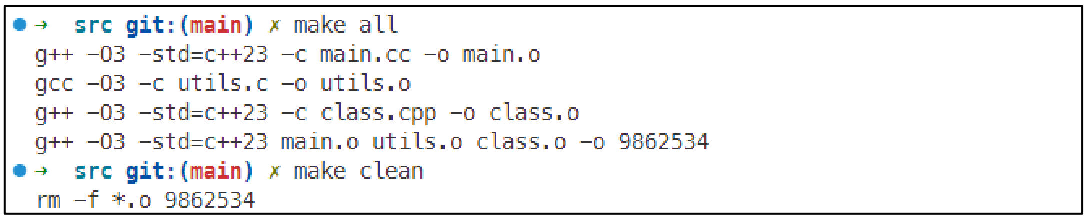
3. We will compile your program by using a Makefile on the workstation. Your Makefile should include at least the following two commands:

   (1) make all

   This command will automatically compile your source code and generate the corresponding objects and an executable file named <student_id> in "src". Otherwise, your program will be not graded.

   (2) make clean

   This command will automatically delete all objects and executable files generated by **make all**.



```
● ➜  src git:(main) ✗ make all
  g++ -O3 -std=c++23 -c main.cc -o main.o
  gcc -O3 -c utils.c -o utils.o
  g++ -O3 -std=c++23 -c class.cpp -o class.o
  g++ -O3 -std=c++23 main.o utils.o class.o -o 9862534
● ➜  src git:(main) ✗ make clean
  rm -f *.o 9862534
```

4. We don't restrict the report format and length. In your report, you have to at least include:
    (1) How to compile and execute your program (You can use screenshot to explain)
    (2) The completion of the assignment (If you complete all requirements, just specify all)
    (3) Algorithms and data structures
    (4) Perturbation strategy and operations
    (5) Cost function and how to determine whether to move to next state or not
    (6) The hardness of this assignment and how you overcome it
    (7) Any suggestions about this programming assignment?

Grading (Total 110%)

The grading is as follows:

(1) Correctness of your placement results (30%)
- ➢ 2 public + 3 hidden test cases

(2) Quality of your placement results (40%)
- ➢ 2 public + 3 hidden test cases

(3) Readability of your code (5%)

(4) The report (10%)

(5) Demo session (25%)

Please submit your assignment on time. Otherwise, the penalty rule will apply:

- Within 24hrs delay: 20% off
- Within 48hrs delay: 40% off
- More than 48hrs: 0 point

Contact

For all questions about PA3, please send E-mail to TA 曹寓恆 (adcarry00334@gmail.com) or come to E1-359 in the time as listed.

- Thu. 17:00-18:30 (04/18, 04/25, 05/02, 05/09)
- Mon. 19:00-20:00 (05/13)
- Fri. 16:00-17:00 (05/24)

Reference

[1] F. Balasa and K. Lampaert, "Symmetry within the sequence-pair representation in the context of placement for analog design," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 7, pp. 721-731, July 2000, doi: 10.1109/43.851988.

[2] Stacked MOSFETs in analog layout | Pulsic

[3] Yun-Chih Chang, Yao-Wen Chang, Guang-Ming Wu, and Shu-Wei Wu. 2000. B*-Trees: a new representation for non-slicing floorplans. In Proceedings of the 37th Annual Design Automation Conference (DAC '00). Association for Computing Machinery, New York, NY, USA, 458–463. https://doi.org/10.1145/337292.337541

[4] Murata, H., Fujiyoshi, K., Nakatake, S., Kajitani, Y. (2003). Rectangle-Packing-Based Module Placement. In: Kuehlmann, A. (eds) The Best of ICCAD. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-0292-0_42

[5] Tang, Xiaoping, Ruiqi Tian, and D. F. Wong. "Fast evaluation of sequence pair in block placement by longest common subsequence computation." Proceedings of the conference on Design, automation and test in Europe. 2000.

[6] Balasa, Florin. "Modeling non-slicing floorplans with binary trees." IEEE/ACM International Conference on Computer Aided Design. ICCAD-2000. IEEE/ACM Digest of Technical Papers (Cat. No. 00CH37140). IEEE, 2000.

[7] Lin, Po-Hung, Yao-Wen Chang, and Shyh-Chang Lin. "Analog placement based on symmetry-island formulation." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28.6 (2009): 791-804.

[8] KLayout download page

[9] HSPICE User Guide: Simulation and Analysis

[10] Optimize Options (Using the GNU Compiler Collection (GCC))

[11] Multiplier 和 Finger 的区别和优劣讨论_finger 和 multiplier 的区别-CSDN 博客