Andy, Yu-Guang Chen

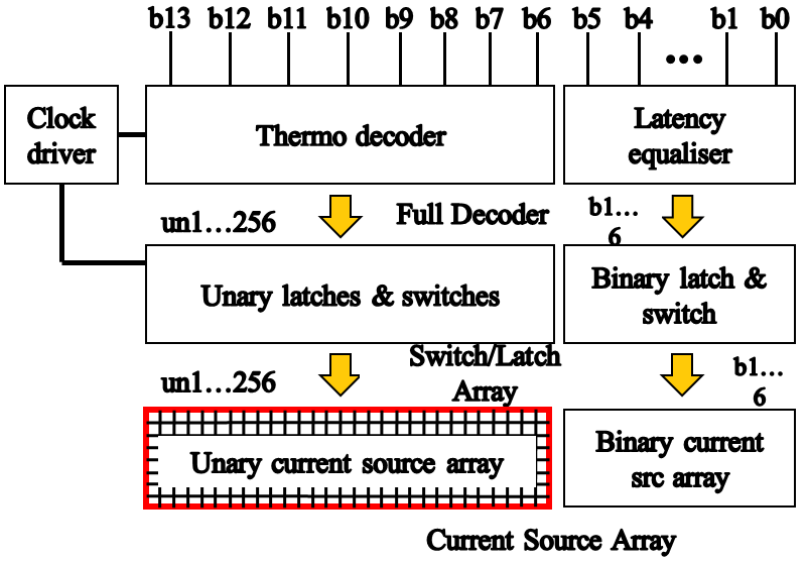
EE6094 CAD for VLSI Design
Programming Assignment 4: Analog Floorplan
(Due: 23:59:59, 2025/06/12)

- Version 1: 2025.05.01 11:30:00

Introduction

The performance of analog circuits heavily relies on the symmetry of devices, as mismatches among theoretically identical components can significantly degrade functionality. Achieving precise matching through symmetrical design is critical for optimal performance. For instance, a 14-bit current-steering digital-to-analog converter (DAC), a type of analog circuit, depends on this symmetry to ensure accurate current outputs across its components, directly impacting its precision and overall performance in converting digital signals to analog.

In this assignment, you are asked to implement an analog placer for solving the rectangle packing problem. The goal is to place given rectangular modules without overlap while minimizing the bounding box area and considering Integral Nonlinearity (INL).



Rectangle Packing Problem in this assignment is defined as follows

Input: Given a set of rectangular modules(transistors)

Output: A legal floorplan result

Objective: Minimize the bounding box area while considering Integral Nonlinearity (INL).

Integral nonlinearity

Integral nonlinearity is a key metric in analog placement that measures the deviation of a device's ideal linear response due to systematic mismatches caused by process variations, thermal gradients, and mechanical stress.

If process variation is evaluated based on the squared distance between each block's center and the centroid of all blocks, then a layout with widely spread and irregularly placed blocks, as seen in Figure 1, will introduce larger deviations, increasing the INL value. On the other hand, a well-organized, symmetric, and compact floorplan, like the one in Figure 2, minimizes these deviations, leading to a lower INL value and better process uniformity.

Figure 1

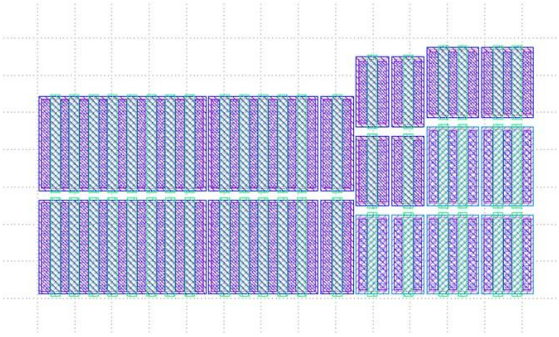


Figure 2

Integral nonlinearity computation

1. Determine its boundary coordinates (left, right, top, bottom) of each block
   - X_{min}: the smallest x-coordinate among all block left edges
   - X_{max}: the largest x-coordinate among all block right edges
   - Y_{min}: the smallest y-coordinate among all block bottom edges
   - Y_{max}: the largest y-coordinate among all block top edges
2. Compute the centroid $(\boldsymbol{X_C, Y_C})$ of the bounding rectangle using:

$$\boldsymbol{X_C} = \frac{X_{min}+X_{max}}{2},\ \boldsymbol{Y_C} = \frac{Y_{min}+Y_{max}}{2}$$

3. Compute the Squared Distance for Each Block
   - For each block $\boldsymbol{B_k}$, calculate its squared Euclidean distance from the centroid:

$$d_k^2 = (x_k - X_c)^2 + (y_k - Y_c)^2$$

   This represents the deviation of each block from the overall layout center.
4. Sort Blocks by Name in Ascending Order
   - Arrange all blocks based on their names in ascending order.
   - This ensures a consistent and structured sequence for cumulative calculations and regression fitting.
5. Accumulate the Squared Distances in Sorted Order
   - Define the cumulative sum $\boldsymbol{S_{actual}(n)}$ of squared distances up to block $\boldsymbol{n}$:

$$S_{actual}(n) = \sum_{k=1}^{n} d_k^2$$

where blocks are processed in the sorted order.

6. Fit a Regression Line to the Cumulative Sum Data
   - Perform linear regression on $(n, S_{ideal}(n))$, where n represents the index of blocks in sorted order and S_n is the cumulative sum.
   - The fitted line equation:

     $$S_{ideal}(n) \approx an + b$$

     where a and b are regression coefficients.
7. Calculate INL Based on Deviation from Ideal Line
   - Define the Integral Nonlinearity as the maximum deviation of actual cumulative values S_n from the regression line:

     $$INL = max| S_{actual}(n) - S_{ideal}(n) |$$

     This measures how much the cumulative process variation deviates from an ideal linear distribution, reflecting nonuniformity in the layout.

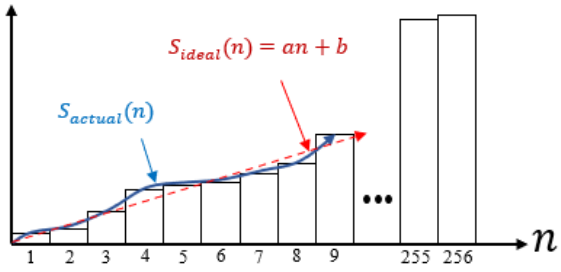


Figure 3

Example

Input files

➢ Building block file (.block)

```
MM0 (4.99 2.12 4 1)
MM1 (4.99 2.12 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

<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.

Output files

- Placement result file (.output)

```
72.4548
4.99 14.52
7.53
MM0 0.0 12.4 (4.99 2.12 4 1)
MM1 0.0 10.28 (4.99 2.12 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: Chip area (Round to four decimal places)
  - (2) Line 2: Chip width and chip height (Round to two decimal places)
  - (3) Line 3: INL (Round to two decimal places)
  - (4) Line 4 to end:
    <device_name> <x> <y> (<width> <height> <col_multiple> <row_multiple>)

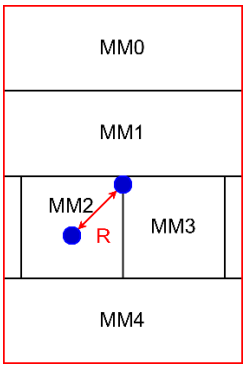


Figure 4

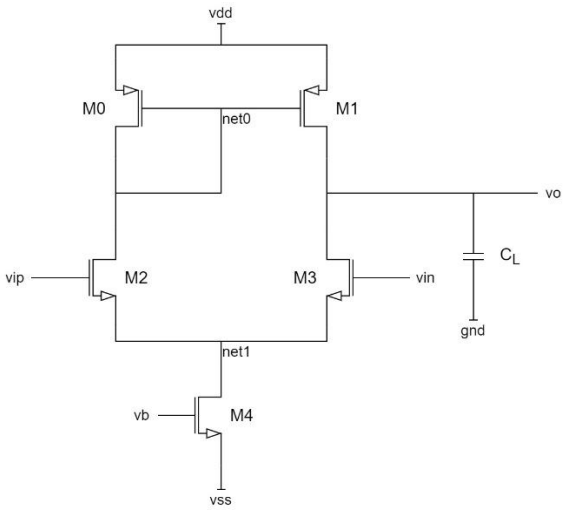


Figure 5

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 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 Area and aspect ratio (Total 5%)

We select two 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.

$$cost = (ChipArea) \times \big(1 + f(AR)\big)\ , \quad F(AR) = \begin{cases} 2(0.5 - AR)\ , & 0 \le AR < 0.5 \\ 0, & 0.5 \le AR \le 2 \\ (AR - 2), & AR > 2 \end{cases}$$

Scores are assigned based on the ranking of all individuals sorted in ascending order of cost, ranging from 0 to 5 points.

- **Minimize chip area**

  Minimize the bounding box area that frames all devices.

- **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:

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

Part 2 INL (Total 5%)

- **Integral nonlinear (INL)**

  Minimize the value of INL.

Scores are assigned based on the ranking of all individuals sorted in ascending order of INL, ranging from 0 to 5 points.

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 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 4.8.5 (GCC)

2. The directory structure inside should follow this format:

```
9862534_PA4/                    (Project folder)
|— 9862534_PA4.cpp              (Main source file)
|— 9862534_PA4_report.pdf       (Report file - only .pdf is accepted)
|— Makefile                     (Compilation script)
|— inc/                         (Header files, if needed)
|   |— file1.h
|   |— file2.h
|— src/                         (Source files, if needer)
|   |— file1.cpp
|   |— file2.cpp
```

3. Compilation & Execution Command

(1) Compile: This command compiles your code and generates the executable file.

```
$make all
```

(2) Execute: This command runs the executable with the provided input file and the required library.

```
$make run input=<input_file> output=<output_file>
```

(3) Clean: This command will automatically remove all the objects and executable file generated by make all and all txt file generated by make run.

```
$make clean
```

(4) We will evaluate your code based on the Makefile you submit. Ensure your Makefile compiles and runs correctly, otherwise you will lose points for both correctness and performance.

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 100%)

The grading is as follows:

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

(2) Quality of your placement results (10%)
- 2 public + 3 hidden test cases

(3) Readability of your code (10%)

(4) The report (20%)

(5) Demo session (30%)

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 PA4, please send E-mail to TA 陳坤民 (qq34415666@gmail.com)

Reference

[1] 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

[2] 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.

[3] 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.