National Taiwan University April 5, 2023
Graduate Institute of Electronics Engineering Handout #7
Physical Design for Nanometer ICs, Spring 2023 Yao-Wen Chang

Programming Assignment #2: Fixed-outline Floorplanning (due 5pm, April 30, 2023 Sunday on-line)

ATTENTION: All programming submissions will be subject to duplication-checking with our database consisting of all submissions from previous and current PD classes; those with ≥ 40% similarity will be penalized. (Most earlier students with this problem eventually failed this course.)

Modified from Problem #1 of the 2003 IC/CAD Contest (Source: Springsoft/Synopsys); please see also Exercise 10.8 of the W&C&C book (pages 628-630).

1. Problem Statement

This programming assignment asks you to write a fixed-outline chip floorplanner to handle hard macros. Given a set of rectangular macros and nets, the floorplanner places all macros within a rectangular chip without overlaps. We assume that the lower-left corner of this chip is the origin (0,0), and no space (channel) is needed between two different macros. The objective is to minimize the area of the chip bounding box and the total net wirelength. The total wirelength W of a set N can be computed by

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

where n_i denotes a net in N, and $HPWL(n_i)$ denotes the half-perimeter wirelength of n_i. The objective of this problem is to minimize

$$Cost = \alpha \frac{A}{A_{norm}} + (1-\alpha)\frac{W}{W_{norm}}$$

where A denotes the bounding-box area of the floorplan, A_{norm} is the average area, W is the total wire length, W_{norm} is the average wire length, and α, $0 \leqq \alpha \leqq 1$, is a user-defined ratio to balance the final area and wirelength. To compute A_{norm} and W_{norm}, we can perturb the initial solution or m times to obtain m floorplans and compute the average area A_{norm} and the average wire length W_{norm} of these floorplans. The value m is proportional to the problem size. Note that a floorplan that cannot fit into the given outline is not acceptable.

2. Input/Output Specification

Input Format

```
Outline: <outline width, outline height>
NumBlocks: <# of blocks>
NumTerminals: <# of terminals>

<macro name> <macro width> <macro height>
… More macros

<terminal name> terminal <terminal x coordinate> <terminal y coordinate>
… More terminals
```

Each test case has two input files, *input.block* and *input.nets*. The first file (*input.block*) gives the outline size, the number of blocks, and the number of terminals defined in this file. Then the block dimensions are listed, followed by the terminal locations. The file format is as follows:
The second file gives the number of nets in the floorplan, followed by the terminal information for each net. The file format is as follows:

```
NumNets: <# of nets>
NetDegree: <# of terminals in this net>
<terminal name>
… More terminal names

… More "NetDegree" and "terminal name"
```

The user-defined ratio α is given through the command-line argument. It ranges between 0 and 1.

Output Format

The output file (*output.rpt*) records the problem output. This report consists of six parts: (1) the final cost, (2) the total wirelength, (3) the chip area, (4) the chip width and height, (5) the runtime in seconds, and (6) the bounding-box coordinate for

```
<final cost>
// Cost = αA + (1-α)W
<total wirelength>
// W = Σ_{n_i∈N} HPWL(n_i)
<chip_area>
// area = (chip_width) * (chip_height)
<chip_width> <chip_height>
//resulting chip width and height
<program_runtime>
//report the runtime in seconds
<macro_name>    <x1>    <y1>    <x2>    <y2>
<macro_name>    <x1>    <y1>    <x2>    <y2>
// (x1, y1): lower-left corner, (x2, y2): upper-right corner
… More macros
```

each macro (specified by the lower-left corner and upper-right corner). The report file format is shown above.

3. Command-line Parameters

The executable binary file must be named as **"fp"** and use the following command format:

```
./fp [α value] [input.block name] [input.net name] [output file name]
```

For example, if you would like to run your binary for the input file **input.block** and **input.nets** with **α = 0.5** and generate a solution named **output.rpt**, the command is as follows:

```
./fp 0.5 input.block input.nets output.rpt
```

4. Example

Figure 1 illustrates an example of the IO files (assume α = 0.5):

Input files (input.block):

```
Outline: 120 120
NumBlocks: 4
NumTerminals: 0
A    40    50
B    60    50
C    60    50
D    40    50
```

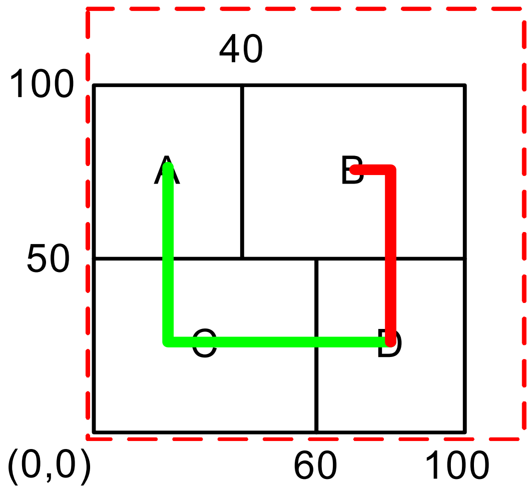

Fig. 1: A floorplanning problem and its solution

(input .nets)

```
NumNets: 2
NetDegree: 3
A
C
D
NetDegree: 2
B
D
```

```
5085
170
10000
100  100
0.24
A    0     50    40    100
B    40    50    100   100
C    0     0     60    50
D    60    0     100   50
```

Output files (output.rpt)

5. Language/Platform

- Language: C/C++
- Platform: Linux. **Please also develop your programs on the servers in the EDA Union Lab.**

6. Advanced Features

Provide a GUI (Graphic User Interface) to show the floorplanning result.

7. Submission:

You need to create a directory named <student id>_pa2/ (e.g. r11943000_pa2/) which must contain the following materials:

- A directory named src/ containing your source codes: only *.h, *.hpp, *.c, *.cpp are allowed in src/, and no directories are allowed in arc/;
- A directory named bin/ containing your executable binary named fp;
- A makefile name makefile or Makefile that produces an executable binary from your source codes by simply typing "make": the binary should be generated under the directory <student id>_pa2/;
- A text readme file named readme.txt describing how to compile and run your program.
- A report named report.pdf on the data structures used in your program and your findings in this programming assignment.

8. Grading Policy

This programming assignment will be graded based on the (1) **correctness** of the program, (2) **solution quality**, (3) **running time**, (4) **report.doc**, and (5) **readme.txt**. Please check these items before your submission. Note that a floorplanning result will not be graded if the running time exceeds one hour on a machine in the EDA Union Lab.

The final score of each case will be first determined by the "temporary score" from our evaluator. We will linearly scale the scores based on the top score for each case. In other words, the top score for each case will be scaled to 10, and others will be scaled according to the top score. You can get your temporary score from the evaluator by the command below:

```
./evaluator <input.block> <input.net> <output.rpt > <alpha>
```

For example, if you want to run the evaluator for the generated solution named output 0.dat with the input file input 0.dat, the command is as follows:

```
./evaluator ami33.block ami33.net output.rpt 0.5
```

The temporary score from the evaluator is generated with the following equation:

$$\text{case score} = \text{quality score} \times 0.8 + \text{runtime score} \times 0.2,$$

where your quality score and runtime score here are scored by interpolation with the tables of quality and runtime results from the submissions of the class in Spring 2022 (pd22), respectively. The top score in pd22 is graded as 9.5, the bottom score as 4, and the others will be graded with even distribution.

9. Online Resources

Sample input files (ami33.block/ami33.nets), readme.txt, and report.doc can be found at the NTU COOL course link below:

https://cool.ntu.edu.tw/courses/25972/assignments/152707

For any questions, please email Cheng-Yu at frankchiang@eda.ee.ntu.edu.tw. Thank you so much for your cooperation. Have fun with this programming assignment!