CS6135 VLSI Physical Design Automation

Homework 4: Global Routing

Due: 23:59, December 21, 2025

1. Introduction

Routing is a crucial step in IC design. In the routing stage, we need to connect all placed cells with wires properly. The routing problem can be divided into two steps: global routing and detailed routing. During global routing, nets are connected on a coarse-grain grid graph with capacity constraints. Then detailed routing follows the solution obtained in global routing to find the exact routing solution. The quality of global routing affects timing, power and density in the chip area, and thus global routing is a very important stage in the design cycle.

In this homework, you are asked to implement an existing algorithm or develop your own algorithm to solve the global routing problem with a set of 2-pin nets.

2. Problem Description

(1) Input:

- A routing region that has been partitioned into an array of rectangular grids called global bins, as known as G-cells, as shown is **Figure 1(a)**. Note that the routing region can be also represented by a grid graph G shown in **Figure 1(b)**, where each vertex $v \in V$ represents a global bin, and each edge $e \in E$ corresponds to a boundary between two adjacent global bins.
- Capacities of global edges.
- A set of nets, $N = \{n_1, n_2, \dots, n_k\}$, to be routed over the grid graph $G = (V, E)$, where each net $n_i, 1 \leq i \leq k$, has a set of pins to be connected, and each pin is at the grid that contains the pin.

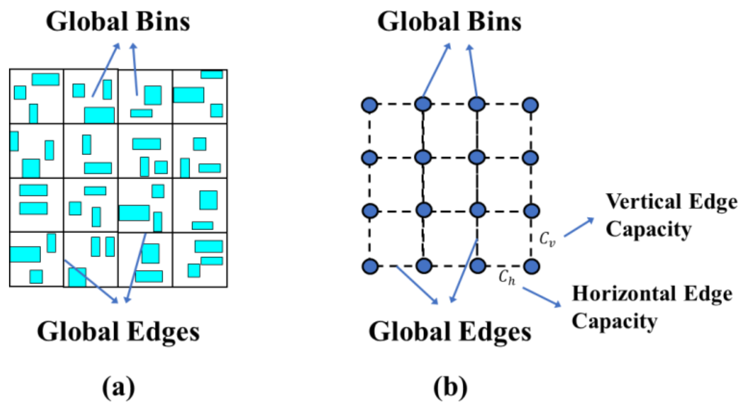


Figure 1: Routing grid and its corresponding grid graph

(2) Output:

The coordinates of critical points on the routing path for each net after global routing is done. Notice that "critical points" means the starting point, ending point and turning points.

(3) Objective:

The routing problem for a net n_i is to find an additional subset of vertices, $s_i \subseteq V$, and a subset of edges, $E_i \subseteq E$, to form a Steiner tree $T_i = (V_i, E_i)$, where $V_i = n_i \cup s_i$. The supply $s(e)$ of an edge e represents the number of available routing tracks passing through it. The number of wires that utilize an edge e is called the demand $d(e)$ on the edge. That is, the demand $d(e)$ represents how many nets that pass through e. The overflow on an edge e is defined to be the difference between its demand and supply as shown below:

$$overflow(e) = \begin{cases} d(e) - s(e), \text{if } d(e) > s(e) \\ 0, \text{otherwise} \end{cases} \tag{1}$$

The primary optimization objective of global routing is to minimize the total overflow on all edges as shown in (2), and the second objective is to minimize the total wirelength on all edges as shown in (3).

$$\min\left(\sum_{\forall e \in E} overflow(e)\right) \tag{2}$$

$$\min\left(\sum_{\forall n_i \in N} \sum_{\forall e \in E_i} length(e)\right) \tag{3}$$

This homework asks you to write a global router that can route multiple nets. To simplify the global routing problem, we have some simplifications as follows:

(a) Consider the routing region on a plane with two routing directions (horizontal and vertical).

(b) Consider only grid-based coordinates. The coordinates of the grid at the lower left corner of the global routing region is (0, 0). As the result, $length(e)$ will always be 1.

(c) Consider only 2-pin nets.

3. Input File

(1) The *.txt* file:

The .txt file specifies the input information. Here is an example:

```
Grid 3 3
// Grid number of columns (x-axis) number of rows (y-axis)
Capacity 2 2
// Capacity horizontal edge capacity vertical edge capacity
NumNets 3
// NumNets number of nets
Net N1 2
// Net net name number of connected pins
Pin P1 0 1
// Pin pin name pin x-grid coordinate pin y-grid coordinate
Pin P2 1 1
Net N2 2
Pin P3 0 2
Pin P4 1 1
Net N3 2
Pin P5 2 2
Pin P6 1 0
```

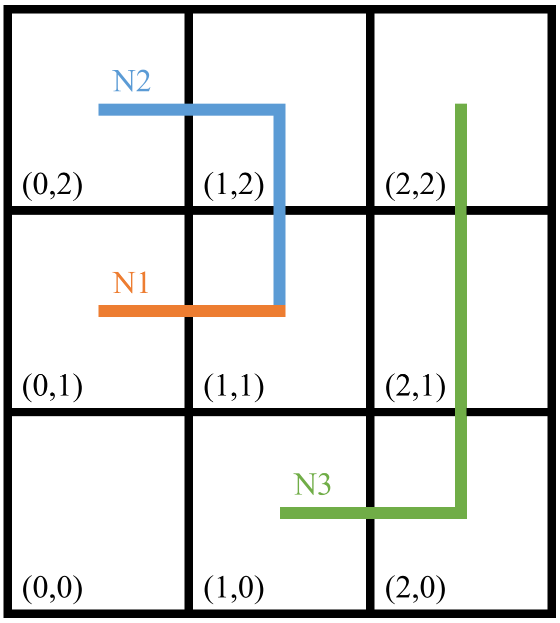
4. Output File

(1) The *.out* file:

The final routing results. The routing result of each net should be a simple path. Here is an example:

```
Wirelength 6
// Wirelength wirelength
Net N1
// Net net name
Segment 0 1 1 1
// Segment x1 y1 x2 y2
Net N2
Segment 0 2 1 2
Segment 1 2 1 1
Net N3
Segment 2 2 2 0
Segment 2 0 1 0
```

Example (assuming each pin is at the center of the grid containing it):



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) The total overflow, wirelength, and runtime of each testcase. Paste the screenshot of the result of running the **HW4_grading.sh**.

(3) Describe the details of your implementation. You may use flowcharts and/or pseudocode to illustrate your algorithm. Explain which routing methods you adopted (e.g., Lee algorithm, A*-search routing, …) and provide a clear description of how each method was implemented.

(4) Did you implement rip-up and re-route?

- ✓ If yes, explain how you determined which nets to rip up and the order in which they were re-routed.
- ✓ If not, describe what methods you used to reduce routing overflow instead.

(5) 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 the TAs to verify.

(3) `bin/` contains your executable file.

(4) `CS6135_HW4_${StudentID}_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_114000000.tar.gz HW4/
```

8. Grading

- ✓ 50%: Achieve zero overflow in public cases to get full points for each case.
- ✓ 30%: Solution quality. The solution quality is evaluated based on the total overflow followed by the total wirelength for each hidden case. There are four hidden cases in total. For each hidden case, the score is computed using the following formula:

  $\text{Hidden Case Score} = 7.5 \times (\max(x) - \text{self}(x))/(\max(x) - \min(x))$

  where:
  - $\textbf{self}(\mathbf{x})$ is the ranking of your solution,
  - $\textbf{max}(\mathbf{x})$ is the maximum ranking of all submissions, and
  - $\textbf{min}(\mathbf{x})$ is the minimum ranking 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 "`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 <input file> <output file>
  ```

  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 450 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 **HW4_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] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang, "NTHU-Route 2.0: A Fast and Stable Global Router," in *Proceedings of International Conference on Computer-Aided Design*, 2008.

[2] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "NTHU-Route 2.0: A Robust Global Router for Modern Designs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 12, pp. 1931-1944, 2010.