CS6135 VLSI Physical Design Automation

Homework 5: Global Routing

Due: 23:59, June 24, 2017

1. Introduction

Routing is a crucial step in IC design. The placement stage determines the location of each cell of an IC. Then in the routing stage, we need to connect all the 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, we ask you to implement a simple global router.

2. Problem Description

The global routing problem requires a set of nets, $N = \{n_1, n_2, \ldots, n_k\}$, to be routed over a grid graph $G(V, E)$. A net n_i, $1 \leq i \leq k$, is a set of pins to be connected. Typically the layout is partitioned into rectangular tiles called global bins as shown in Figure 1(a), and each pin is assumed to lie at the center of the tile that contains the pin. In the grid graph G shown in Figure 1(b), each vertex $v \in V$ represents a global bin, and each edge $e \in E$ corresponds to a boundary between two adjacent global bins.

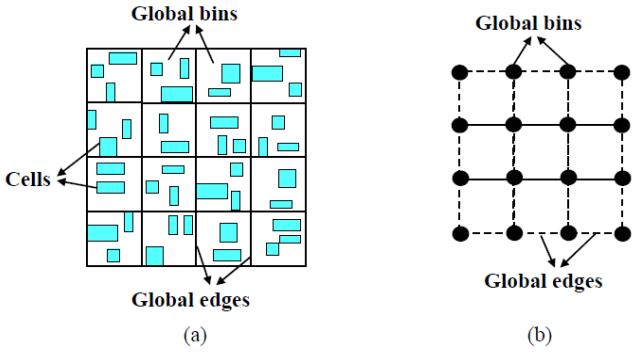


Figure 1. (a) A routing region with 4x4 global bins. (b) The corresponding grid graph of (a)

The routing problem for a net n_i is to find an additional subset of vertices, $v_{i,Steiner} \subseteq V$, and a subset of edges, $e_i \subseteq E$, to form a Steiner tree $t_i = (v_i, e_i)$, where $v_i = n_i \cup v_{i,Steiner}$. 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. In other words, 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:

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

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

$$\min\left(\sum_{e_i \in E} overflow(e_i)\right) \quad (2)$$

This homework asks you to write a global router that can route 2-pin nets. To simplify the global routing problem, we have some simplifications as follows:

(a) Consider only two layers.

(b) Consider only tile-based coordinates. The lower left corner of the global routing region in each layer is $(0, 0)$. The tile width and height are ignored, because all X and Y coordinates are tile-based.

(c) Consider only 2-pin nets.

(d) Consider only fixed wire width and spacing. All wire width and spacing are equal to 1.

3. Input/output specification

Input format:

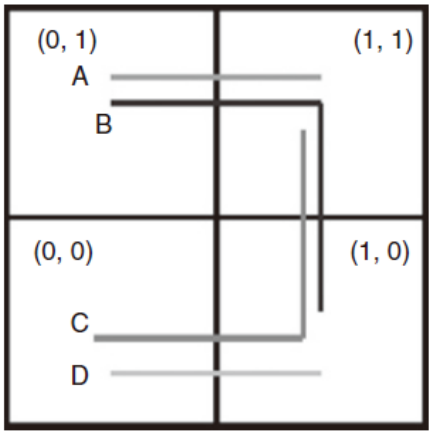
```
grid # #                  // numbers of horizontal tiles, and vertical tiles
vertical capacity #       // vertical capacity
horizontal capacity #     // horizontal capacity
num net #                 // number of nets
net_name net_id number_of_pins
x y
x y
.....  [Repeat for the appropriate number of nets]
```

Output format:

```
net_name net_id
([x11],[y11],[1])-([x12],[y12],[1])
([x21],[y21],[1])-([x22],[y22],[1])
.....
!
```

[Repeat for the appropriate number of nets]

Example:



Input file:

```
grid 2 2
vertical capacity 2
horizontal capacity 2
num net 4
A 0 2
0 1
1 1
B 1 2
0 1
1 0
C 2 2
0 0
1 1
D 3 2
0 0
1 0
```

Output file:

```
A 0
(0, 1, 1)-(1, 1, 1)
!
B 1
(0, 1, 1)-(1, 1, 1)
(1, 1, 1)-(1, 0, 1)
!
C 2
(0, 0, 1)-(1, 0, 1)
(1, 0, 1)-(1, 1, 1)
!
D 3
(0, 0, 1)-(1, 0, 1)
!
```

4. Language/Platform

(a) Language: C/C++

(b) Platform: Linux (NTHUCAD)

5. Requirements

(a) Your executable file name should be named as route.

(b) **You must use the following command format to execute your program:**

```
./route <input_file_name> <output_file_name>
```

(c) Your program must terminate **within 10 minutes** for each released test case.

(d) **Please make sure your output files can be verified by our provided checker.** The checker shall tell you the related problems about your global routing results. You should use the following command line to run the verifier:

```
./verify <input_file_name> <output_file_name>
```

6. Required Files

Please tar all the required files as **CS6135_HW5_${StudentID}.tar.gz** and submit it to the iLMS. The file structure is almost the same as HW2 and HW3.

(1) **src/** contains all your sources code, your **Makefile** and **README**.

- **README** must contain how to compile and execute your program. An example is like the one shown in HW2.
- **Makefile** must contain "make all" and "make clean"

(2) **output/** contains all your outputs of testcases for comparison.

(3) **bin/** contains your compiled executable file. Note that your generated executable must be named as route.

(4) **CS6135_HW5_${STUDENT_ID}_report.pdf** contains your report. Put it where **CS6135_HW5_sepc.pdf** is.

The file structure would be like the one shown in HW2.

You can use the following command to compress your directory on a workstation:

```
$ tar -zcvf CS6135_HW5_{StudentID}.tar.gz <directory>
```

For example: `$ tar -zcvf CS6135_HW5_105062901.tar.gz HW5/`

7. Report

Your report should contain the following content, and you can add more as you wish. The more the better.

(1). A cover page containing the title, your name, and your student ID

(2). How to compile and execute your program, and give an execution example.

(3). The total overflow, wirelength and run time for each test case.

(4). The details of your algorithm. You could use flow chart(s) and/or pseudo code to help elaborate your algorithm. If your method is similar to some previous works/papers, please cite the papers and reveal your difference(s).

(5). The details of your implementation. What tricks did you do to speed up your program or to enhance your solution quality?

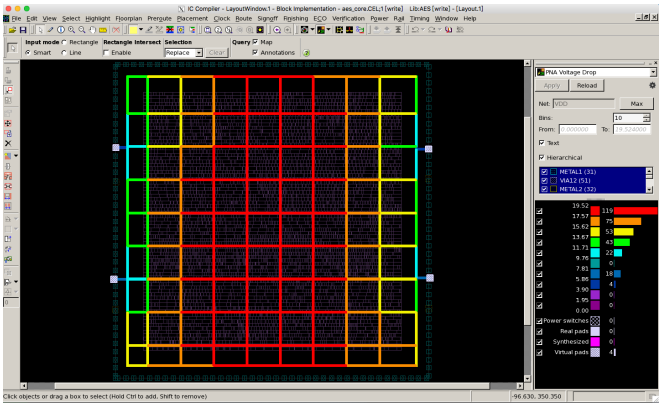
(6). Please compare your results with the top 3 students' results from last year and show your advantage either in runtime or in solution quality. Are your results better than theirs?

- ✓ If so, please express your advantages to beat them.
- ✓ If not, it's fine. If your program is too slow, then what could be the bottleneck of your program? If your solution quality is inferior, what do you think that you could do to improve the result in the future?

(7). What have you learned from this homework? What problem(s) have you encountered in this homework?

7. Bonus

- ✓ A Makefile to build and delete your program by `make` and `make clean`.
- ✓ Additionally, for each testcase, you could draw the congestion map like the following example.



8. Grading

- ✓ 80%: The solution quality and runtime of each testcase, hidden testcases included
- ✓ 20%: The completeness of your report
- ✓ 5%: Bonus