# Lab2 – DP based global routing

Deadline: 5/8/2023 23:59

## Problem Description

Implement a global router that can complete ISPD 2008 Global Routing Contest [1], and try to minimize the total overflow (TOF), max overflow (MOF), wirelength (WL) and runtime.

## Input Format (Parser is provided)

All benchmarks have multiple metal layers (three-dimensional), with additional congestion information. The input file format specifies the size and shape of the global routing graph, the capacity on edges of the graph, the number of signal nets, and the number of pins and their positions within each net.

Format with comments in red (these will not be in actual input files). This example illustrates a problem with five routing layers.

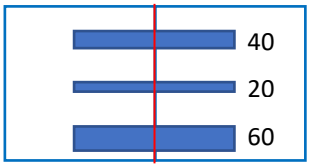
grid # # # (x grids, y grids, number of layers)
vertical capacity # # # # # (vertical capacity by default on each layer)
horizontal capacity # # # # #
minimum width # # # # #
minimum spacing # # # # #
via spacing # # # # #
lower_left_x lower_left_y tile_width tile_height

num net #
netname id_# number_of_pins minimum_width
x y layer
x y layer
...
[repeat for the appropriate number of nets]

# capacity adjustments (to model congestion)
column row layer column row layer reduced_capacity_level
[repeat for the number of capacity adjustments]

Description:

- In the vertical and horizontal capacity lines, the first number indicates the capacity on the first layer, the second number indicates the second layer, and so on. Minimum wire widths and minimum wire spacings are also specified; this impacts capacity calculations.
- The lower left corner (minimum X and Y) of the global routing region is specified, as well as the width (X dimension) and height (Y dimension) of each tile.
- Pin positions are given as XY locations, rather than tile locations. Conversion from pin positions to tile numbers can be done with $\text{floor}(\text{pin\_x} - \text{lower\_left\_x}) / \text{tile\_width}$ and $\text{floor}(\text{pin\_y} - \text{lower\_left\_y}) / \text{tile\_height}$. Pins will not be on the borders of global routing cells, so there should be no ambiguity. All pins will be within the specified global routing region.
- Each net will have a minimum routed width; this width will span all layers. When routing, compute the utilization of a global routing graph edge by adding the widths of all nets crossing and edge, plus the minimum spacing multiplied by the number of nets. Each wire will require spacing between its neighbors; think of this as having one-half of the minimum spacing reserved on each side of a wire.




If the value of minimum spacing in this layer is 10, then the demand of red grid edge is $40 + 20 + 60 + 10 + 10 + 10 = 150$

- Congestion is modeled by including capacity adjustments. Pairs of (adjacent) global routing tiles may have a capacity that is different from the default values specified at the start of a benchmark file.
- Capacity specified in the 2nd and 3rd lines is a measure of the *available space*, not the number of global routing tracks. If the minimum wire width is 20, the minimum spacing 10, and the capacity of a tile is given as 450, this corresponds to 15 minimum width tracks (15 * (20 + 10)).
- If a net has more than 1000 sinks (i.e., #pins > 1000), you don't have to route it.
- If all pins of a net fall in the same tile, you don't have to route it.

Output Format (handled by given layer assignment algorithm)

netname id_#
(x1,y1,layer1)-(x2,y2,layer2)
[repeat for the appropriate number of segments in nets]
!
[repeat for the appropriate number of nets]

- Only one of {x,y,z} of two endpoints in segments can be different.
- The {x,y} must be given as XY locations, the conversion from tile numbers to XY locations can be done with $\text{tile\_x} \times \text{tile\_width} + \text{lower\_left\_x}$ and $\text{tile\_y} \times \text{tile\_height} + \text{lower\_left\_y}$.

Sample input file and output file with illustrations

Input: https://www.ispd.cc/contests/08/3d.txt
Output: https://www.ispd.cc/contests/08/3ds1.txt
Description(pdf): https://www.ispd.cc/contests/08/3d.pdf

Benchmarks

We will run adaptec1, adaptec2, adaptec3, adaptec4, adaptec5, bigblue1, bigblue2, bigblue3, newblue1, newblue2, newblue5, newblue6 and two hidden benchmarks. These benchmarks will be available in newE3 soon.

Router Evaluation and Grading

For each benchmark, the results of all students are compared with those of a golden router RCE [2] in terms of TOF, MOF, WL, and runtime. A less TOF a router has, a higher rank it has. If there is a tie in TOF, MOF is used to break the tie. If MOF is still the same, the routed wire length and runtime are the 2nd tie breaker. The exact quality metric for 2nd tie-breaker is the

$$WL \times (1 + 0.04 \times \log_2(runtime \,/\, median\_runtime))$$

If the rank you get in a benchmark is higher than or equal to the golden router, you get 100 point in that benchmark. If not, the TOF of the golden router $TOF_{golden}$ is used to evaluated your score in that benchmark.

For overflowed cases $(TOF_{golden} > 0)$.

| $TOF_{yours}$ region | Score region |
|---|---|
| $[1.0,1.2) \times TOF_{golden}$ | $[95,99)$ |
| $[1.2,1.4) \times TOF_{golden}$ | $[90,95)$ |
| … | … |
| $[4.8,\infty) \times TOF_{golden}$ | $[0,5)$ |

For overflow-free cases $(TOF_{golden} = 0)$

| $TOF_{yours}$ region | Score region |
|---|---|
| $[0,10)$ | $[95,99)$ |
| $[10,20)$ | $[90,95)$ |
| … | … |
| $[190,\infty)$ | $[0,5)$ |

The final total score will be the weighted sum of each score in each benchmark, the weights are all 1 by default, but are subject to change.

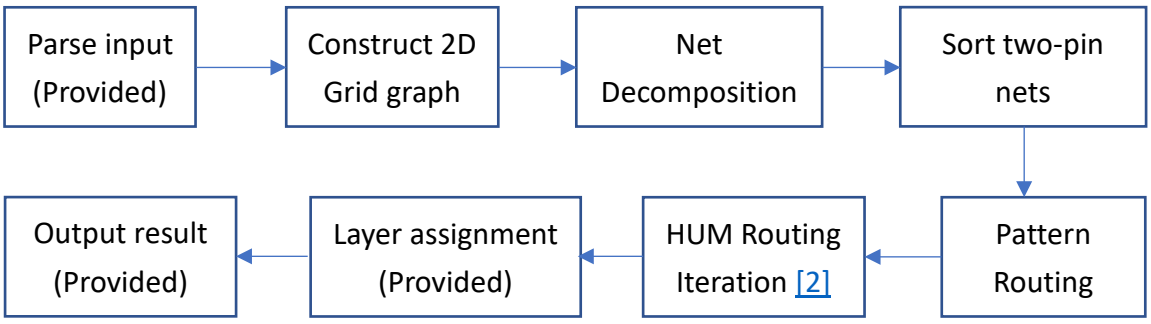
## Evaluation Script

You can use this perl script provided from [1] to check net connectivity and get TOF, MOF, WL of a routing solution.

## Provided Files

```
Lab2/
├── main.cpp          Sample code that can generate 3ds1.txt
├── ispdData.h        Data structure for ISPDParser
├── LayerAssignment.h Part of layer assignment code of NCTUGR [3]
├── LayerAssignment.cpp Part of layer assignment code of NCTUGR [3]
├── Makefile          Sample Makefile
├── eval2008.pl       The evaluation script
└── 3d.txt            The sample input
```

## Suggested Flow



## Submission

Zip all the source codes and a Makefile in <student_id>.zip without any folders like below, then submit the .zip file to E3 by the deadline.

```
<student_id>.zip/
├── main.cpp
├── Makefile
├── <Any source codes and necessary files>
└── <We will run your code here>
```

- Late submission: Score * 0.95 before 5/15, Score * 0.8 after 5/15 and before 5/22. After 5/22, you will get 0 point.
- Hard coding the output based on input is not allowed.
- Any work by fraud will absolutely get 0 point.

## Executing Procedure

1. unzip <student_id>.zip
   - If fail → 0 point in all benchmarks
2. make
   - If "Makefile" not found or compile error → 0 point in all benchmarks
3. ./router <inputFile> <outputFile>
   - If "router" is not found or cannot be executed → 0 point in all benchmarks
   - If runtime limit (default 30mins) is reached → 0 point in this benchmark
   - If the command returned non-zero exit code (unhandled C++ exceptions, segmentation fault, out of memory, …) → 0 point in this benchmark
4. perl eval2008.pl <inputFile> <outputFile>
   - If <outputFile> is not found or it's corrupted → 0 point in this benchmark
   - If the command returned non-zero exit code (open nets, diagonal routed nets, disjointed nets, …) → 0 point in this benchmark

## Appendix

Testing environment

- OS: CentOS 8
- CPU: Intel(R) Xeon(R) Gold 5220R CPU @ 2.20GHz 48 Cores 96 Threads
- Memory: 256G
- GCC version: 8.3.1

[2] results on [1]

| Benchmarks | TOF | MOF | WL | Runtime (sec) |
|---|---|---|---|---|
| adaptec1 | 0 | 0 | 5820849 | 28.3 |
| adaptec2 | 0 | 0 | 5407474 | 21.68 |
| adaptec3 | 0 | 0 | 13626514 | 52.59 |
| adaptec4 | 0 | 0 | 12310042 | 42.42 |
| adaptec5 | 0 | 0 | 16619509 | 89.09 |
| bigblue1 | 0 | 0 | 6499329 | 47.43 |
| bigblue2 | 28 | 2 | 9497076 | 57.3 |
| bigblue3 | 0 | 0 | 13372234 | 50.23 |
| newblue1 | 492 | 2 | 4810928 | 48.92 |
| newblue2 | 0 | 0 | 7663497 | 25.57 |
| newblue5 | 0 | 0 | 24233695 | 117.96 |
| newblue6 | 0 | 0 | 19324723 | 87.07 |

References

[1] https://www.ispd.cc/contests/08/ispd08rc.html

[2] W. -H. Liu, Y. -L. Li and C. -K. Koh, "A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing," 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2012, pp. 713-719.

[3] W. -H. Liu, W. -C. Kao, Y. -L. Li and K. -Y. Chao, "NCTU-GR 2.0: Multithreaded Collision-Aware Global Routing With Bounded-Length Maze Routing," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 5, pp. 709-722, May 2013, doi: 10.1109/TCAD.2012.2235124.