2009 IC/CAD Contest

Problem B3:

Obstacle-Avoiding Rectilinear Clock Routing with Preferred Directions

Source: Synopsys

1. Introduction

Finding a minimum route for an n-pin net can be modeled as a minimum Steiner tree problem of n points. To match the realistic routing patterns better, routers usually use the Rectilinear Steiner Tree (RST) to avoid slant wires. Besides, a router needs to deal with obstacles on multiple metal layers (masks) for chip routing. Those obstacles are usually formed by some Macro cells or pre-routed power/ground nets. Therefore, the current routing problem can be simplified by finding a minimum rectilinear Steiner tree with the presence of obstacles in multiple layers.

Considering signal integrity and IC manufacturing, we normally assign a preferred direction (PD) to each metal layer such that all edges on the given metal layer are either horizontal or vertical. Moreover, the routing directions of some specific routing ranges, named "switch direction", need to be perpendicular to the preferred direction of the located metal layer. **In real world IC design flow, the "switch direction" region is manually added in designs to increase the routability. It is usually added among lots of obstacles. For example, the power/ground wire directions may be different in the top design from those in hard macros. We need to specify switch direction regions over such kind of hard macros.** Therefore, Preferred-Direction Rectilinear Steiner Tree (PDRST) is the most practicable and preferable.

Although a real router needs to deal with more and more complicated DRC (Design Rule Checking), DFM (Design for Manufacture) rules and timing constraints simultaneously, we only practice to deal with a skew issue in addition to the total length in this contest.

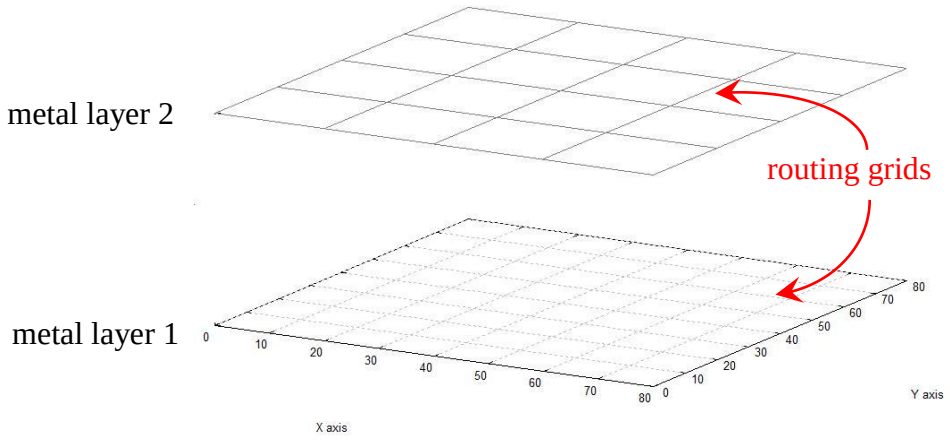
2. Problem Description

Given a set of nets (each net contains a set of pins) and a set of rectangle obstacles on multiple metal layers; the problem is:

(1) To find RSTs connecting all pins for each net with minimum total routing cost. The definition of 'total routing cost' is the summation of the net-costs of each net. The definition of 'net-cost" of a net is the summation of the total wire length and the skew of the net. The definition of 'skew' of a net is the difference between the maximum delay and the minimum delay from the source pin to all sink pins among the same net. The delay from the source pin to a sink pin is a constant (different metal/via layers may have different constants) times the wire length from the source pin to the sink pin.

(2) All edges of RSTs must satisfy the preferred directions of metal layers and the switch directions of specific routing ranges without any OPEN, SHORT, or SAME-NET LOOP. **The switch directions of specific routing ranges are perpendicular to the preferred direction of the located metal layer. Note that the edges on the boundaries of these specific routing ranges can be horizontal and vertical.**

(3) This is a grid-based routing problem. **It means that the routing grids are pre-determined and all the edges must be on the grids. The bottom-most and left-most grids of each layer are located in X- and Y-axes, respectively.** The spacing of the routing grids is determined by the routing pitch (different metal layers may have different pitches). **An illustration of the routing grids is as follows:**



A via is used to connect between two points of the same X- and Y- coordinates on adjacent metal layers. Legal positions for vias are the intersections of the grids on adjacent metal layers.

The total number of nets won't exceed 10000. **The total number of pins for each net won't exceed 20.** The chip dimension won't larger than (10000000, 10000000). The total number of metal layers is within 15.

3. Input/Output Format

We will detail input and output formats, total routing cost calculation, and finally summarize with an example.

3.1 Input

The input file is an ASCII text file and consists of six parts: chip dimension, information of metal layers, information of routing ranges with switch directions, information of via layers, information of nets, and information of obstacles.

The first row specifies the chip dimension and is defined by two points *i*, *j* – the coordinates of the bottom-left $(x_i\ y_i)$ and the top-right points $(x_j\ y_j)$, as shown in the following.

```
.chip (0 0) (1000 800)
```

The keyword ".layer *n*" signifies the beginning of the definition of the information of metal layers. The number *n* denotes the total number of metal layers. Each statement consists of the serial number of metal layers (1 to *n*), the preferred direction (H/V for horizontal/vertical direction), **the routing pitch (the spacing of the routing grids),** the **unit-grid** delay with preferred/switch directions, and the **unit-grid** delay with non-preferred/non-switch directions for each metal layer. The following is an example with five metal layers.

```
.layer 5
1 H 10 30 3000
2 V 10 30 3000
3 H 10 30 3000
4 V 20 80 8000
5 H 40 200 20000
```

The keyword ".switch *n*" signifies the beginning of the definition of the routing ranges with switch directions. The number *n* denotes the total number of the routing ranges with switch directions. **Each statement describes a routing range with switch directions.** Each routing range with switch directions is defined by two points i, j – the coordinates of the bottom-left $(x_i\ y_i\ z_i)$ and the top-right points $(x_j\ y_j\ z_j)$; where z_i must be the same as z_j (to be on the same metal layer). A routing range with switch directions can be on any area, any location, and any metal layer. The following example defines two routing ranges with switch directions.

```
.switch 2
(100 90 2) (300 600 2)
(20 40 5) (40 100 5)
```

The keyword ".via *n*" signifies the beginning of the definition of the information of via layers. The number *n* denotes the total number of via layers. **Vias on via layer *n* are used to connect routes between metal layers *n* and *n+1*.** Each statement consists of the serial number (1 to *n*), the equivalent wire length, and the **unit-via** delay for each via layer. The following is an example with four via layers.

```
.via 4
1 10 30
2 10 30
3 20 80
4 40 200
```

The keyword ".net *n*" signifies the beginning of the definition of the nets. The number *n* denotes the total number of nets. Each statement consists of the net name, the total number of pins, the coordinate of its source pin, and the coordinates of its sink pins. The following is an example with three nets.

```
.net 3
Net1 5 (400 560 1) (640 720 5) (810 90 2) (120 700 3) (440 200 4)
Net2 3 (450 540 1) (860 660 3) (240 160 5)
Net3 4 (200 110 2) (940 800 1) (440 540 4) (870 660 3)
```

The keyword ".obs *n*" signifies the beginning of the definition of the information of rectangle obstacles. The number *n* denotes the total number of rectangle obstacles. **Each statement describes a rectangle obstacle.** Each rectangle obstacle is defined by two points *i*, *j* – the coordinates of the bottom-left $(x_i\ y_i\ z_i)$ and the top-right points $(x_j\ y_j\ z_j)$; where z_i must be the same as z_j (to be on the same metal layer). A rectangle obstacle can be on any area, any location, and any metal layer. The following example defines three rectangle obstacles.

```
.obs 3
(100 100 1) (500 200 1)
(500 250 2) (550 750 2)
(320 280 5) (800 520 5)
```

3.2 Output

The output file is an ASCII text file and consists of three parts: information of routing solution, the total wire length, and the total routing cost.

The keyword ".net *net_name n*" signifies the beginning of the definition of the information of routing solution. The *net_name* denotes the net name from the input file. The number *n* denotes the total number of routes (edges). Each route is defined

by two points *i*, *j* – the coordinates of the bottom-left $(x_i\ y_i\ z_i)$ and the top-right points $(x_j\ y_j\ z_j)$. All routes in the output should be a horizontal route, a vertical route, or a via. For example (180 610 1) (190 620 1) is not acceptable, as it is diagonal. Remember that each route should be different only in X-, Y-, or Z- coordinates. The following is a routing result of the net Net2 with ten routes.

```
.net Net2 10
(240 160 5) (240 160 4)
(240 160 4) (240 160 3)
(240 160 3) (240 160 2)
(240 160 2) (240 160 1)
(240 160 1) (450 160 1)
(450 160 1) (450 160 2)
(450 160 2) (450 660 2)
(450 540 1) (450 540 2)
(450 660 2) (450 660 3)
(450 660 3) (860 660 3)
```

The keyword ".wirelength *n*" signifies the total wire length of the routing solution. The number *n* denotes the total wire length.

```
.wirelength 2130
```

The keyword ".routingcost *n*" signifies the total routing cost of the routing solution. The number *n* denotes the total routing cost.

```
.routingcost 7830
```

3.3 The Calculation of Routing Cost

Assume the input contains n nets. The routing result of net n_i $(1 \le i \le n)$ contains r_i routes. Two terminals of route $r_{i,j}$ $(1 \le j \le r_i)$ are $(x_1^{r_{i,j}}\ y_1^{r_{i,j}}\ z_1^{r_{i,j}})$ and $(x_2^{r_{i,j}}\ y_2^{r_{i,j}}\ z_2^{r_{i,j}})$. The unit-grid delay of route $r_{i,j}$ is $C_{r_{i,j}}$. Net n_i contains p_i sink pins and the set of $E_{p_{i,j}}$ is the path from the source pin to the sink pin $p_{i,j}$ $(1 \le j \le p_i)$. Then, the total wire length and the total routing cost can be calculated as follows.

1. Total wire length:

$$total\_wirelength = \sum_{\substack{1 \le i \le n \\ 1 \le j \le r_i}} l_{r_{i,j}},$$

$$\boldsymbol{l_i = |x_1^i - x_2^i| + |y_1^i - y_2^i| + |z_1^i - z_2^i| \times L_j,}$$

where L_i denotes the equivalent wire length of via layer *i*.

2. Total routing cost:

$$total\_routingcost = total\_wirelength + \sum_{1 \le i \le n} (max\_delay_i - min\_delay_i),$$

$$max\_delay_i = \max_{1 \le j \le p_i} \left(\sum_{\substack{1 \le k \le r_i \\ r_{i,k} \in E_{p_{i,j}}}} \left(l_{r_{i,k}} / S_m * C_{r_{i,k}} \right) \right),$$

$$min\_delay_i = \min_{1 \le j \le p_i} \left(\sum_{\substack{1 \le k \le r_i \\ r_{i,k} \in E_{p_{i,j}}}} \left(l_{r_{i,k}} / S_m * C_{r_{i,k}} \right) \right),$$

where S_i denotes the routing pitch (the spacing of the routing grids) of metal layer *i*. Note that the skew of 2-pin net is 0.

3.4 Example of input and output formats

We give a test case for example. The example is a design with 1 net (contains 3 pins), 1 rectangle obstacle, 1 routing range with switch directions, 2 metal layers, and 1 via layer.

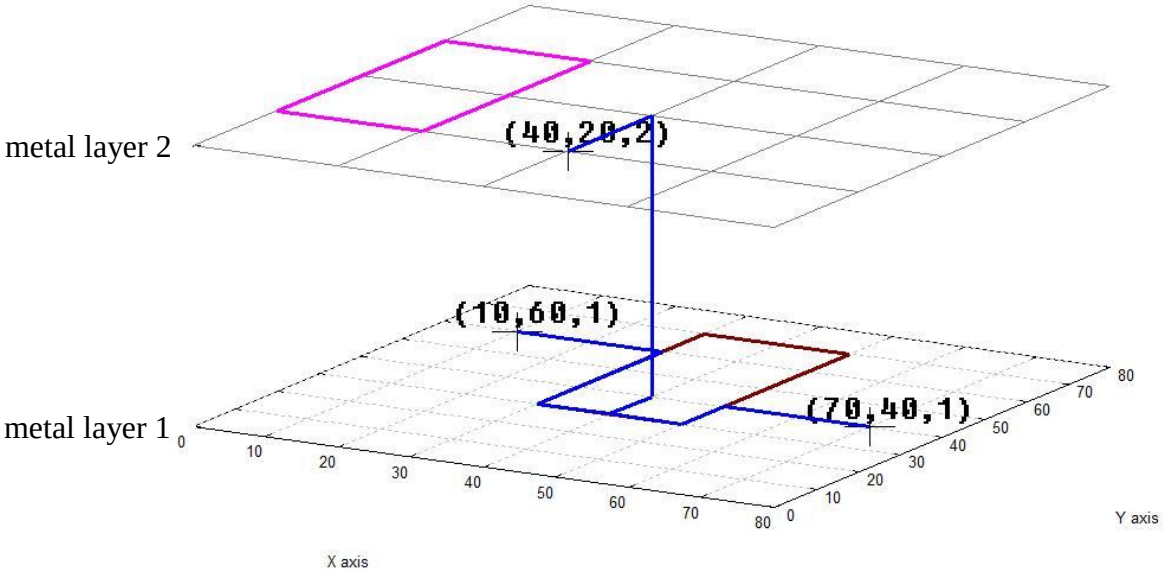
```
.chip (0 0) (80 80)
.layer 2
1 H 10 30 3000
2 V 20 80 8000
.switch 1
(30 30 1) (50 70 1)
.via 1
1 20 40
.net 1
Net1 3 (40 20 2) (10 60 1) (70 40 1)
.obs 1
(0 20 2) (20 60 2)
```

The output is a routing result with 9 routes. The total wire length and total routing cost are 150 and 210, receptively.

```
.net Net1 9
(10 60 1) (30 60 1)
```

(30 30 1) (30 60 1)
(30 30 1) (40 30 1)
(40 30 2) (40 40 1)
(40 40 1) (40 40 2)
(40 20 2) (40 40 2)
(40 30 1) (50 30 1)
(50 30 1) (50 40 1)
(50 40 1) (50 70 1)
.wirelength 150
.routingcost 210

An illustration of the routing result is as follows:



4. Language/Platform

- Language: C or C++ metal layer 1
- Platform: Linux or Sun OS/Solaris

The binary should be called as

./pdrst ($input_file_name) ($input_file_name).out

For example, some input file is named ‘data1’; the corresponding output file should be ‘data1.out’. The command line is looked like:

./pdrst data1 data1.out

5. Evaluation

Basic Requirements:

- No crash in the execution of program.
- All nets must be routed without any OPEN, SHORT, or SAME-NET LOOP. If a net is with any violations, the case is a failed case (the score of this case is zero).
- All routes are placed on the grids.

Standard Requirements:

- The *total routing cost* of the routing result.

Advanced Requirements:

- Run time will be considerated how to contribute to the final score.
- Memory usage shouldn't exceed 4G bytes.

6. Support

An utility *evaluate* is provided to test your result. It can evaluate the following items:

- The routing result is / is not on the grids.
- The routing result is valid / invalid.
- The total wire length.
- The total routing cost.

The command is as follows:

./evaluate *input_file_name output_file_name*

References

[1] C.-H. Liu, Y.-H. Chou, S.-Y. Yuan, and S.-Y. Kuo, "Efficient multilayer routing based on obstacle-avoiding preferred direction Steiner tree," *in Proc. of ACM International Symposium on Physical Design*, pp. 118--125, April 2008.

[2] C.-W. Lin, S.-Y. Chen, C.-F. Li, Y.-W. Chang, and C.-L. Yang, "Obstacle-avoiding rectilinear Steiner tree construction based on

spanning graphs," *in IEEE Trans. Computer-Aided Design*, vol. 27, no. 4, pp. 643--653, April 2008.

[3] I. H.-R. Jiang, S.-W. Lin and Y.-T. Yu, "Unification of obstacle-avoiding rectilinear Steiner tree construction," *in Proc. of IEEE International SOC Conference*, pp. 127--130, September 2008.

[4] I. H.-R. Jiang and Y.-T. Yu, "Configurable Rectilinear Steiner Tree Construction for SoC and Nano Technologies," *in Proc. Of IEEE International Conference on Computer Design*, October 2008.

[5] C.-W. Lin, S.-L. Huang, K.-C. Hsu, M.-X. Li, and Y.-W. Chang, “Multi-layer obstacle-avoiding rectilinear Steiner tree construction based on spanning graphs," *in IEEE Trans. Computer-Aided Design*, vol. 27, no. 11, November 2008.