ICCAD 2018 CAD Contest

Obstacle-Aware On-Track Bus Routing

About Liao, Hua-Yu Chang, Owen Chi, and Jane Wang
Synopsys Taiwan Co., Ltd.,
1F., No. 25, Gongye E. 4th Rd., Hsinchu Science Park, Hsinchu, Taiwan

Contents

0. Announcements

September

- 2018-09-14 FAQ updated
- 2018-09-14- Evaluator updated
- 2018-09-11- FAQ updated
- 2018-09-06- Final deadline of B extended to 9/14.
- 2018-09-06- Testcase and evaluator updated.
- 2018-09-06- FAQ Beta Test announced.
- 2018-09-04- FAQ updated
- 2018-09-03- FAQ updated

August

- 2018-08-02- FAQ updated

July

- 2018-07-31- FAQ updated
- 2018-07-27- New example is release. Beta deadline of B extended to 7/29.
- 2018-07-27- FAQ updated
- 2018-07-12- Problem B description is updated e in **Sec. VI** and **Sec.VII**.
- 2018-07-12- Alpha Test announced.
- 2018-07-12- TestCase updated.

June

- 2018-06-19- FAQ updated
- 2018-06-06- FAQ updated

May

- 2018-05-31- FAQ updated
- 2018-05-28- FAQ updated
- 2018-05-14- FAQ updated

April

- 2018-04-25- Testcase is announced
- 2018-04-25- FAQ updated
- 2018-04-12- FAQ updated

ICCAD 2018 CAD Contest

Obstacle-Aware On-Track Bus Routing

About Liao, Hua-Yu Chang, Owen Chi, and Jane Wang
Synopsys Taiwan Co., Ltd.,
1F., No. 25, Gongye E. 4th Rd., Hsinchu Science Park, Hsinchu, Taiwan

I. Introduction

Bus routing in advanced technology node is a challenging task due to: a) non-uniform and complex routing track configuration; and b) need to route in between small obstacles while maintaining the same routing topology for all bus bits.

Routing tracks (tracks) are essential in advanced technology node to help router adhere to design rules and help mask coloring. In order to meet the routing requirements for different buses (e.g. different wire width), uniform track configuration is not sufficient. The tracks in this problem have width constraint which only allows wires with width smaller than or equal to the constraint to route on them, and may not fully cover the whole design. Figure 1 illustrates a simple configuration of three types of tracks. The green tracks have the smallest width constraint, blue tracks have the larger width constraint, and red tracks have the largest width constraint. Also, the green and the red routing tracks cover the whole design from top to bottom, whereas the blue routing tracks only cover partial of the design.

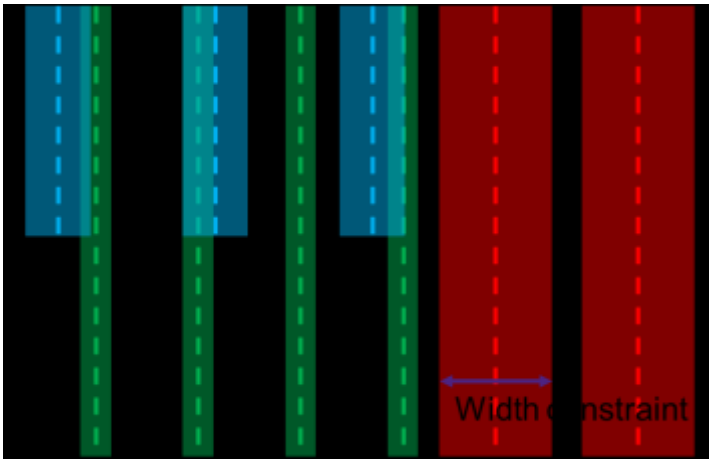


Figure 1 Routing Track Example

Small obstacles such as power vias make bus routing even more challenging. Since such obstacles are scattered throughout certain layers, it is not possible to find continuous routable area if bus bits are not allowed to route in between some of them. Figure 2 shows an example of routing a bus on track through small obstacles while

maintaining same routing topology for all bus bits. The dotted lines are tracks, the red rectangles are obstacles, the orange rectangles with bolded outlines are pins, and the orange thick lines are routed paths.

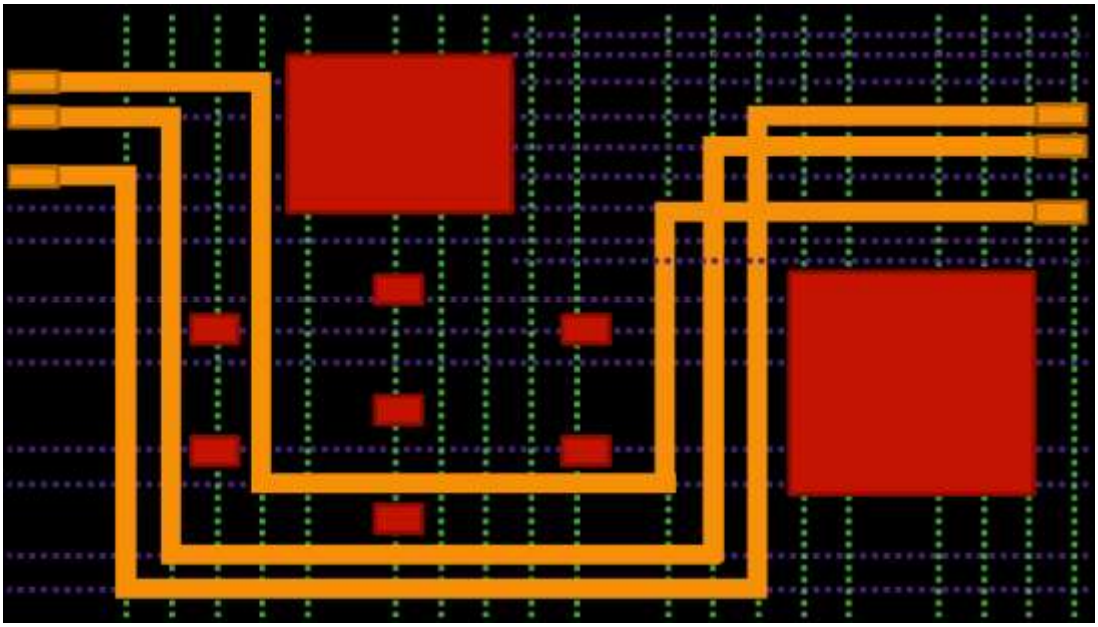


Figure 2 Bus Routing Example

II. Problem Formulation

Given the following inputs:

- A design boundary $\boldsymbol{BD}$.
- A set of layers $\boldsymbol{L} = \{L_1, L_2, ..., L_k\}$, where layer L_i, $1 \leqq i \leqq k$, defines
  - Routing direction $\boldsymbol{D_i}$.
  - Spacing $\boldsymbol{S_i}$.
- A set of tracks $\boldsymbol{T} = \{T_1, T_2, ..., T_m\}$, where track T_i, $1 \leqq i \leqq m$, defines
  - Track width $\boldsymbol{WT_i}$.
- A set of buses $\boldsymbol{B} = \{B_1, B_2, ..., B_n\}$, where bus B_i, $1 \leqq i \leqq n$, defines
  - Number of bits $\boldsymbol{NB_i}$.
  - Number of pin shapes for each bit $\boldsymbol{NP_i}$.
  - Pin shapes for bit j, $1 \leqq j \leqq NP_i$, $\boldsymbol{P_{ij}} = \{P_{ij1}, P_{ij2}, ..., P_{ijNPi}\}$.
  - Bus width for each layer $\boldsymbol{WB_i} = \{WB_{i1}, WB_{i2}, ..., WB_{ik}\}$.
- A set of obstacles $\boldsymbol{O} = \{O_1, O_2, ..., O_u\}$.

The program should output a set of routing paths $p = \{p_1, p_2, ..., p_v\}$ such that p connects all pin shapes of all buses. Each path can be either a horizontal wire, a vertical wire, or a via connecting two layers.

Take Figure 2 for example, the input can be broken down into two layers as shown in Figure 3. The input for this design is as follows:

- 2 layers L_1 and L_2.

- 14 tracks on L_1 (green dotted lines) and 17 tracks on L_2 (blue dotted lines).
- 1 set of buses with 3 bits and 2 pin shapes for each bit on L_2 (bolded outline orange rectangles).
- 8 obstacles on L_1 and 9 obstacles on L_2 (red rectangles).

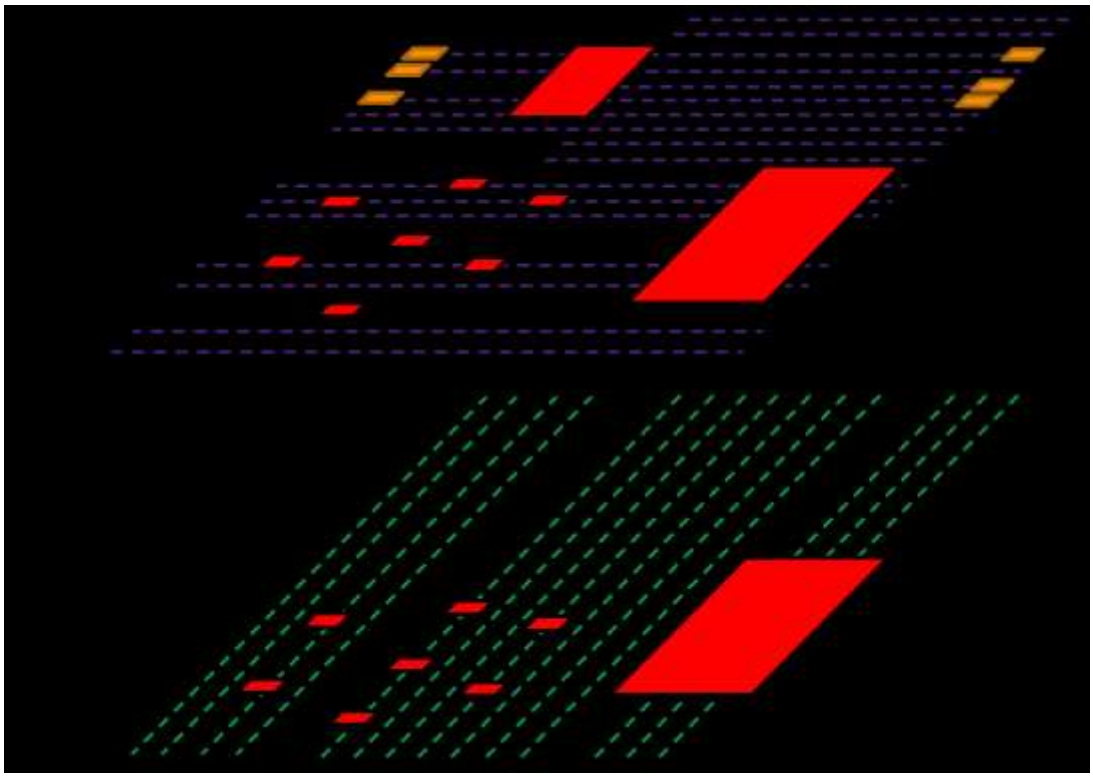


Figure 3 Input Example

The output of Figure 2 can be broken down into two layers as shown in Figure 4. The output is composed of 9 horizontal wires (thick orange rectangles) on L_2, 6 vertical wires on L_1, and 12 vias (thin orange lines).

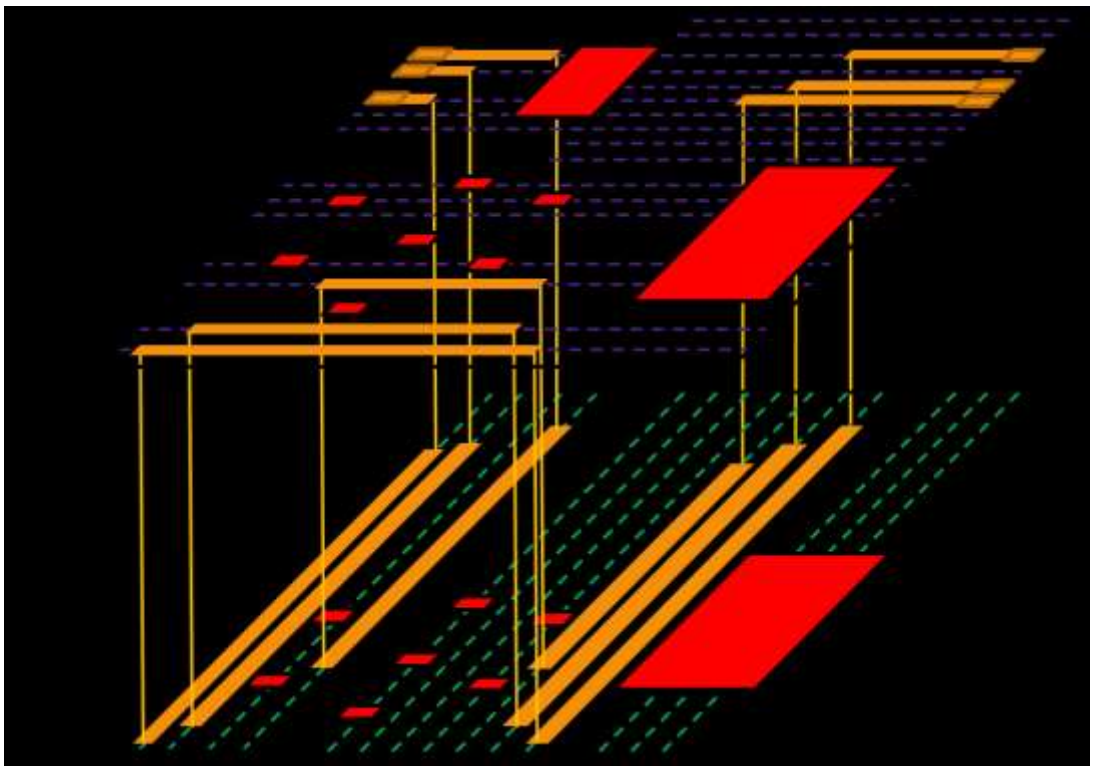


Figure 4 Output Example

III. Input Format Rules

A. Coordinates

The input objects, such as design boundary, pin shapes, tracks, obstacles, etc., are represented by either a point, a line, or a rectangle. The coordinates of the points, lines, and rectangles have the following rules:

- Coordinates are non-negative integers and are smaller than UINT32_MAX.
- A point is expressed by X coordinate followed by Y coordinate enclosed by parenthesis. E.g. (100 500).
- A line is expressed by two points with either the X coordinates or the Y coordinates being the same. E.g. a vertical line (100 500) (100 1000).
- A rectangle is expressed by two points with the first point being the lower left corner of the rectangle and the second point being the upper right corner of the rectangle. E.g. (100 500) (1000 1000).

B. Design boundary

Design boundary is represented by a rectangle. All other input objects are guaranteed to be enclosed by the design boundary. Input objects can touch the design boundary. E.g. design boundary (0 0) (1000 1000) and obstacle (0 100) (50 500). The design boundary has the following input format:

```
DESIGN_BOUNDARY <rectangle>
```

For example:

```
DESIGN_BOUNDARY (0 0) (1000 1000)
```

C. Layers

The layers are stacked from bottom to top based on their input order. They can be connected with neighboring ones through vias. For example, if the layers are L_1, L_2, and L_3, this means L_1 is the bottommost layer, and L_3 is the topmost layer. L_1 can connect to L_2, L_2 can connect to L_1 and L_3, and L_3 can connect to L_2.

Each layer has a routing direction and a spacing constraint. The routing direction is either horizontal or vertical. The tracks or the output routing paths on that layer must follow its routing direction. Note that neighboring layers do *NOT* guarantee to have different routing directions. The spacing constraint specifies the distance to which the output routing paths must keep with the obstacles, the design boundary, and other routing wires on that layer.

The layers have the following input format:

```
LAYERS <number of layers>
    [<layer name> <direction> <spacing>]...
ENDLAYERS
```

For example:

```
LAYERS 3
L1 horizontal 50
L2 vertical 100
L3 horizontal 80
ENDLAYERS
```

D. Tracks

The track is represented by a line on a layer with a width constraint. As mentioned in the previous section, the direction of the track is always the same as the routing direction of the layer that the track is on. For example, if L_1's routing direction is horizontal, all tracks on L_1 are horizontal.

Each track also comes with a width constraint. The width constraint represents the maximum width that a wire can route on. For example, suppose the width constraint of track T_1 is on L_1 and it width constraint is WT_1, and bus B_1's width constraint on L_1 is WB_{11}, then bus B_1 can only route on T_1 if WB_{11} is smaller than or equal to WT_1. The value of the width constraint is guaranteed to be multiples of 2.

Note that tracks can overlap with each other while having different width constraints. In this case, the output path can route on the track as long its width constraint is smaller than or equal to the maximum width constraints of the overlapping tracks. Take Figure 5 for example, if T_1 is (50 100) (500 100) on L_1 with width constraint $WT_1 = 10$,

and T_2 is (250 100) (1000 100) on L_1 with width constraint WT_2 = 20. In the overlapped segment of the two tracks, (250 100) (500 100), any routing path with width constraint smaller than or equal to 20 can route on it.

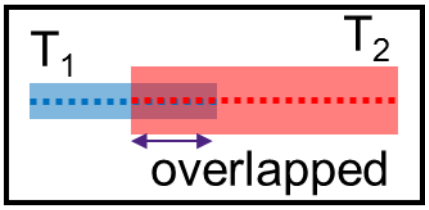


Figure 5 Track Overlap Example

The tracks have the following input format:

```
TRACKS <number of tracks>
    [<layer name> <line> <width constraint>]...
ENDTRACKS
```

For example:

```
TRACKS 4
L1 (0 10) (100 10) 50
L1 (0 20) (100 20) 50
L2 (10 0) (10 100) 100
L2 (20 0) (20 100) 100
ENDTRACKS
```

E. Buses

The bus is represented by the pin shapes of the bits and the width constraint on each layer. The bus has the following rules:

- Each bus specifies its number of bits and number of pin shapes for each bit. For example, for bus B_i, if its number of pin shapes is NP_i, then all bits have NP_i pin shapes.
- The number of bits is at least 1. Note that the number of bits can be 1. In this case, the bus acts like normal signal net.
- The number of pins shapes for each bit is at least 2.
- Each pin shape is represented by a rectangle on a layer. Pin shapes are guaranteed not to overlap with each other.
- The width constraint of the bus on a layer means the output routing paths for the bus on that layer has the width equal to the width constraint. The values of the width constraints of the bus are guaranteed to be multiples of 2.
- Each pin shape is guaranteed to overlap with at least one track which the width constraint of the track is larger than or equal to the bus's width

constraint on that layer. Note that pins may not align with the tracks perfectly. In real life, router needs to tap-off misaligned and off-track pins to nearest track. In this problem, we focus on routing the trunk of the bus, there will be no off-track pins. Also, for misaligned pins, output wires touching the pin shape is considered as connected as shown in Figure 6.

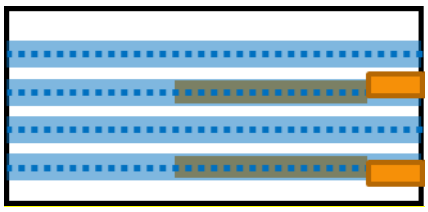


Figure 6 Misaligned Pin Example

The overlapped tracks might not be on the same layer as the pin shape. Take Figure 7 for example, there are 2 horizontal pin shapes on L_1, however, the tracks on L_1 (red tracks) are vertical. If routing started from L_1, the wires of both bits would overlap with each other (using the same red track). In this case, vias can first be placed in the pin shapes and start the routing from L_2 using different blue tracks.

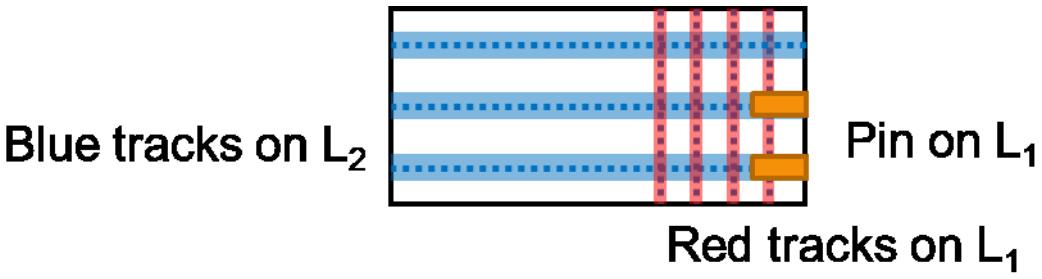


Figure 7 Pin Overlapped Tracks on Different Layers

- The order of the physical locations of the pin shapes of the bits is guaranteed to be either the same as that of the input bit name or the reverse of it (depends on how it is viewed). Take Figure 8 for example, bus B_1 has 3 bits, *b1<0>*, *b1<1>*, and *b1<2>*, and has 3 pin shapes for each bit on the left, bottom, and right of the design. Take the pin shapes on the left for example, their order from top to bottom is *b1<0>*, *b1<1>*, *b1<2>*, which is the same as the input order. (if viewed from bottom to top, it is of the reverse order of the input). For pin shapes on the bottom and right, if viewed from left to right and top to bottom, they are of the reverse order of the input. The pin shapes are guaranteed to be clustered into groups so that their ordering can be

distinguished clearly.

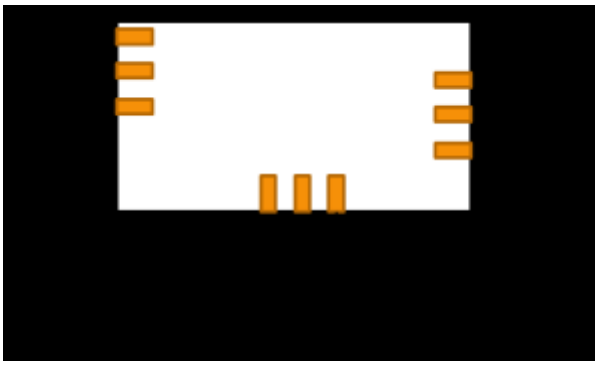


Figure 8 Pin Shape Location Order Example

The buses have the following input format:

```
BUSES <number of buses>
    [BUS <bus name>
    <number of bits>
    <number of pin shapes for each bit>
    WIDTH <number of layers>
        <width constraint>...
    ENDWIDTH
    [BIT <bit name>
        [<layer name> <rectangle>]...
    ENDBIT]...
    ENDBUS]...
ENDBUSES
```

For example:

```
BUSES 1
BUS B1
2
2
WIDTH 3
50
100
80
ENDWIDTH
BIT 0
L1 (0 0) (10 10)
L2 (100 200) (110 210)
ENDBIT
```

```
BIT 1
L1 (0 20) (10 30)
L2 (100 220) (110 230)
ENDBIT
ENDBUS
ENDBUSES
```

F. Obstacles

The obstacle is represented by a rectangle on a layer. Obstacles may overlap with each other.

The obstacles have the following input format:

```
OBSTACLES <number of obstacles>
    [<layer name> <rectangle>]...
ENDOBSTACLES
```

For example:

```
OBSTACLES 3
L1 (40 50) (60 100)
L1 (45 80) (100 200)
L2 (100 200) (120) (230)
ENDOBSTACLES
```

IV. Output Format Rules

The routing paths are written out to a file bus bit by bus bit. Each routing path can be either a horizontal wire on a layer, a vertical wire on a layer, or a via from a layer to a neighboring layer. The output of routing path has the following rules:

- If the routing path is a wire on a layer, it should be output in the format of the name of layer followed by the coordinates of the line. E.g. L1 (20 30) (50 30). The direction of the line should match the direction of the layer. Though the wire is represented by a line, it actually represents a rectangle shape that has width of the bus width constraint on that layer. Take Figure 9 for example, suppose p_1 is a horizontal wire, (20 30) (50 30), for bit $b1<0>$ of bus B_1 on L_1, the actual shape it represents is expanding the line by half the bus B_1 width constraint on L_1, which is $WB_1/2$, on both sides of the line like the figure on the right. Note that the rectangle the wire represents is used to check the spacing constraint with other objects (other wires, obstacles, etc.), not the line itself.

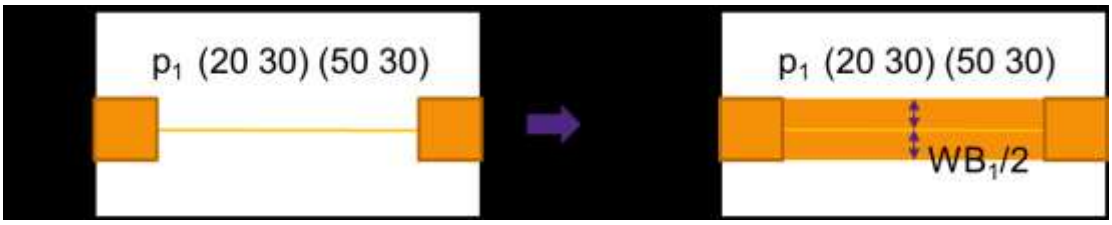


Figure 9 Line to Wire Example

- If the routing path is a via, it should be output in the format of a point and the lower layer of the two neighboring layers that it connects. E.g. L_1 (10 10) represents a via at (10 10) connecting L_1 to L_2. Outputting a via on the topmost layer is illegal.
- Routing paths of the same bit are considered to be connected if they overlap with each other. Take Figure 10 for example, there are 2 horizontal lines, p_1 and p_4, on L_1; 1 via, p_2, from L_1 to L_2, and 1 vertical line, p_3, on L_2. Suppose all of them belong to the same bit, then p_1, p_2, and p_3 are connected, whereas p_4 are not connected with the other three routing paths. Pin shapes touching or overlapping with routing paths are considered as connected. Stacking vias are allowed and are considered as connected. For example, suppose via 1 is at L1 (10 10), and via 2 is at L2 (10 10), then L1 is connected to L3 at point (10 10).

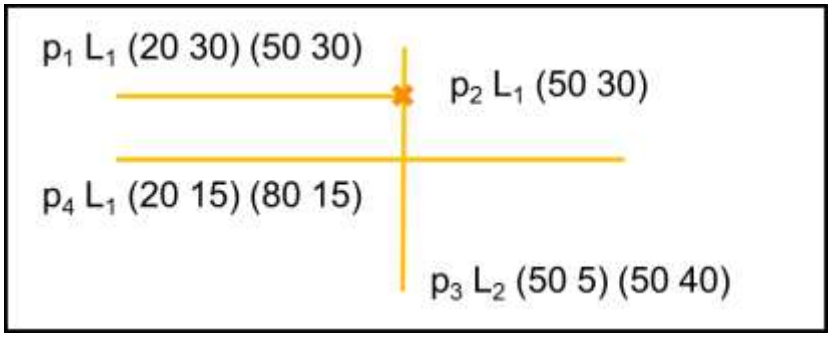


Figure 10 Connectivity Example

- The output order of the routing paths of a bus bit is irrelevant to results. That means, take Figure 10 for example, output order p_1, p_2, p_3, p_4, and p_4, p_3, p_2, p_1 represent the exact same results

The output has the following format:

```
[BUS <bus name>
    [BIT <bit name>
    PATH <number of routing paths>
```

```
        [<layer> <point> | <layer> <rectangle>]...
    ENDPATH
    ENDBIT]...
ENDBUS]...
```

For example:

```
BUS B1
BIT 0
PATH 3
L1 (20 30) (50 30)
L1 (50 30)
L2 (50 5) (50 40)
ENDPATH
ENDBIT
ENDBUS
```

V. Evaluation Methodology

The goal of the program is to complete the bus routing while minimizing the overall cost for each test case. The overall cost for a test case is defined as:

Overall_cost $\boldsymbol{C}$ = routing_cost $\boldsymbol{C_R}$ + penalty_cost $\boldsymbol{C_P}$

The routing cost is to measure the routing quality, such as routing resource used, bus topology, etc., and the penalty cost is to apply penalty to route fail buses and spacing violations between output routing paths. The routing cost only apply to buses that are routed successfully. The route fail buses will induce the penalty cost that is almost always higher than the routing cost if the bus is routed successfully. The contestants will be ranked by the **summation of all overall costs over all test cases**.

A. Route Success and Route Fail

A successfully routed bus has the following 3 requirements:

- Routing paths of a bit connect all pin shapes of the bit.
- All wires are on-track and do not violate the width constraint of the track.
- All bits are routed in the same topology.

The first requirement is straightforward, Figure 11 is an example of an unconnected bus.

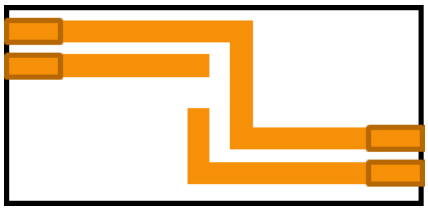


Figure 11 Unconnected Bus Example

For the second requirement, any wire off-track or has bus width constraint larger than the track width constraint is considered as route fail. Figure 12 shows both situations (note that horizontal tracks are not drawn). The one on the left has a vertical wire not on the vertical track. The one on the right has a vertical wire with width larger than the track width. Note that all tracks are within the design boundary, thus, any routing path outside of design boundary is definitely off-track and results in route fail of the bus.

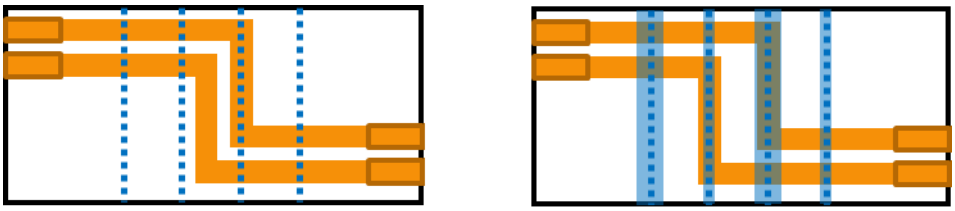


Figure 12 Track Violation Example

The third requirement is more complex to define. Here we break down routed bus into segments, where each segment represents a set of wires of different bits that have the same sequence when traced from a set of pin shapes. Take Figure 13 for example, suppose we start tracing wires from the set of pin shapes on the left, the two immediate horizontal wires (colored blue in the middle figure) connected to the pin shapes form the first segment Seg_1. Now, continue tracing from Seg_1 to the next set of wires, which are the vertical ones that form Seg_2 (the rightmost figure). In this example, this bus has 4 segments.

Figure 13 Bus Segment Example

Upon tracing the segments of the bus, the bits are considered as the same topology if the following four criterions are met:

- All bits should have the same number of wires.
- All wires traced from all bits should have the same layer sequencing. Take Figure 13 for example, suppose horizontal wires are on L_1 and vertical wires are on L_2 and suppose we start tracing the wires from the pin shapes on the left. Wires for both bits have the same layer sequencing: L_1, L_2, L_1, L_2.
- All wires traced from all bits should route towards the same direction. Take Figure 14 for example, the blue segment in the left figure has one bit routing

upward while the other one routing downward. Thus the bits are not considered as same topology. Note that segments can be T-junctions as well. The blue segment in the right figure is a T-junction. So, if one bit in a segment is a T-junction, all bits should to be T-junctions.

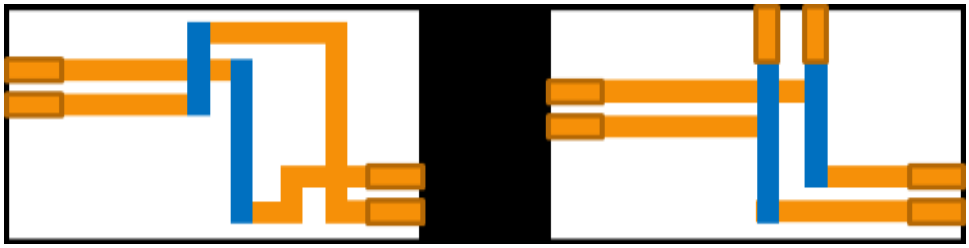


Figure 14 Traced Wire Direction Example

- Within each segment, the wires of different bits should maintain the same or the reverse order as the order seen from the pin shapes (as discussed in Section III.E). Two examples are given in Figure 15 to show bits routed in different topology and the bus is considered as route fail.

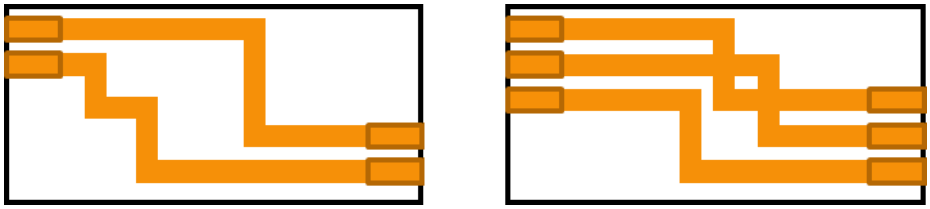


Figure 15 Different Topology Example

B. Routing Cost

If the bus is routed successfully, its routing cost is calculated. The routing cost $\boldsymbol{C_R}$ consists of three factors related to routing quality: the wire length cost $\boldsymbol{C_w}$, the segment cost $\boldsymbol{C_s}$, and the compactness cost $\boldsymbol{C_c}$. It is defined as:

$$\boldsymbol{C_R} = \sum_i^{All\ buses} \alpha \cdot \boldsymbol{C_{wi}} + \beta \cdot \boldsymbol{C_{si}} + \gamma \cdot \boldsymbol{C_{ci}}, \text{ where}$$

$$\boldsymbol{C_{wi}} = \frac{\sum_j^{All\ bits\ of\ bus\ i} \frac{wire\ length\ of\ bit\ j}{half\ parameter\ wire\ length\ of\ bit\ j}}{\#bits\ of\ bus\ i}$$

$$\boldsymbol{C_{si}} = \frac{\#segments\ of\ bus\ i}{lower\ bound\ of\ \#segments\ of\ bus\ i}$$

$$\boldsymbol{C_{ci}} = \frac{\sum_j^{All\ segments\ of\ bus\ i} \frac{width\ of\ segment\ j}{lower\ bound\ width\ of\ segment\ j}}{\#segments\ of\ bus\ i}$$

The wire length of a bit is calculated by summing up the length of all wires of that bit. The wire length cost for the bit is a normalized cost calculated by dividing the summed up length by the half parameter wire length of the bit. The wire length cost of a bus is the sum of the wire length cost of all bits divided by the number of bits. Thus, if

the bus is routed with minimum length required, the wire length cost of the bus should be close to 1. Take Figure 16 for example, taking detours will cause the wire length cost go up.

Figure 16 Wire Length Cost Example

The segment cost of a bus is defined as the number of segments of the bus divided by the lower bound number of segments of the bus. The lower bound is determined by the evaluation program based on the layer of the pin shapes and the location of the obstacles, to estimate how many segments are needed to route the bus. If the bus is routed with minimum number of segments, the segment cost of the bus should be close to 1.

Figure 17 Segment Cost Example

The compactness cost of a segment is defined as the width of the segment divided by the lower bound width of the segment. If the direction of the segment is horizontal, the width of the segment is defined as the Y coordinate of topmost wire in the segment minus the Y coordinate of the bottommost wire in the segment. If the direction of the segment is vertical, the calculation is between the X coordinate of the rightmost and the leftmost wire.

The lower bound width of the segment is minimum width needed for placing all wires on that layer as compact as possible (without considering the tracks). For example, if S_1 is 50, NB_1 is 3, and WB_1 is 30, then the lower bound width of the segment on L_1 for B_1 is 160 since the minimum distance between wires is 80 (spacing plus width) and there are 3 bits. An exception is the segment connected to the pin shapes. The lower bound width for such segments is the distance between the topmost and bottommost pin shapes if the direction of the layer is horizontal, or is the distance between the

rightmost and leftmost pin shapes if the direction of the layer is vertical.

The compactness cost for a bus is the summation of the compactness cost of all segments divided by the number of the segments. If all segments are routed with widths close to the lower bound, the compactness cost for the bus should be close to 1.

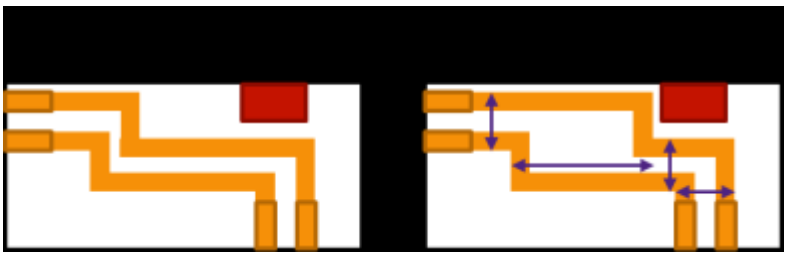


Figure 18 Compactness Cost Example

Overall, in an ideal case, the routing cost of a perfectly routed bus is close to $\alpha + \beta + \gamma$.

C. Penalty Cost

The penalty cost C_P consists of spacing violation penalty P_s and route fail penalty P_f. It is defined as:

$C_P = P_s + P_f$, where

P_s = number of spacing violations $\times$ δ

P_f = number of route fail buses $\times$ ε

For each wire, it should keep at least the distance of the spacing constraint of the layer with the following objects:

- Other wires of different bits.
- Obstacles.
- Design boundary.

Take Figure 19 for example, p_1 should keep at least S_1 with p_2, O_1, and the design boundary. Spacing violations between the two objects will only be counted once. For example, if p_1 did not keep enough spacing with the left edge and the bottom edge of the design boundary, the spacing violation between the two is counted once. Note that the routing paths of the route fail buses still count toward spacing violations if there are any. Contestants should remove floating or failed routing paths. Any violation caused by the floating or failed routing paths counts toward the cost as well. Vias are treated as zero size and will not have spacing violation with other objects. Depending on the values of δ and ε, if a routed bus has too many spacing violations, it may not be worthwhile to leave it routed.

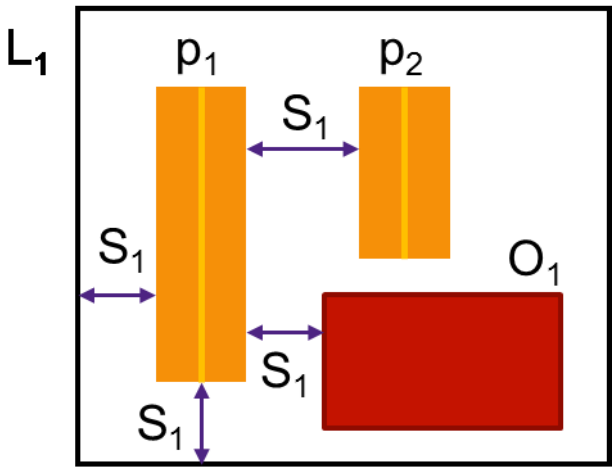


Figure 19 Spacing Violation Example

The rules of determining route success or route fail of a bus is already defined in Section V.A. If a bus route fail, the route fail penalty ε is added to the penalty cost. Generally, ε is large enough (much larger than $\alpha+\beta+\gamma$) such that the penalty of a failed bus is almost always larger than the routing cost of the reasonably routed bus. Note that large test cases do not guarantee to have a clean solution with no bus fail or violations. Also, sometimes a result with few violations might have smaller cost than that of one with no violations. Contestants need to determine how to optimize the routing results based on the given cost function.

D. Runtime Limit

Each test case has a runtime limit specified in the input file as **wall clock time** in **minutes**. Programs that exceed the runtime limit will be killed. No matter the program is killed or terminates normally, the output file will be evaluated. If no output file is found or the output format is wrong, the overall cost for the test case is treated as if all buses route fail.

E. Multithreading

Multithreading is allowed and encouraged. The maximum number of cores that can be used is 4.

VI. Example Test Case and Execution Steps

Take Figure 2 for example, the input is:

```
RUNTIME 1
ALPHA 5
BETA 1
GAMMA 5
DELTA 8
```

```
EPSILON 200
DESIGN_BOUNDARY (0 0) (1000 1000)
LAYERS 2
L1 vertical 20
L2 horizontal 30
ENDLAYERS
TRACKS 34
L1 (100 0) (100 1000) 10
L1 (140 0) (140 1000) 10
L1 (180 0) (180 1000) 10
L1 (220 0) (220 1000) 10
L1 (260 0) (260 1000) 10
L1 (360 0) (360 1000) 6
L1 (400 0) (400 1000) 6
L1 (440 0) (440 1000) 6
L1 (480 0) (480 1000) 6
L1 (520 0) (520 1000) 6
L1 (620 0) (620 1000) 10
L1 (660 0) (660 1000) 10
L1 (700 0) (700 1000) 10
L1 (740 0) (740 1000) 10
L1 (780 0) (780 1000) 10
L1 (880 0) (880 1000) 6
L1 (920 0) (920 1000) 6
L1 (960 0) (960 1000) 6
L2 (0 50) (1000 50) 10
L2 (0 100) (1000 100) 10
L2 (0 200) (1000 200) 10
L2 (0 250) (1000 250) 10
L2 (0 350) (1000 350) 10
L2 (0 400) (1000 400) 10
L2 (0 450) (1000 450) 10
L2 (0 550) (1000 550) 10
L2 (0 600) (1000 600) 10
L2 (0 650) (1000 650) 10
L2 (0 750) (1000 750) 10
L2 (0 800) (1000 800) 10
L2 (450 500) (1000 500) 10
```

```
L2 (450 700) (1000 700) 10
L2 (450 850) (1000 850) 10
L2 (450 900) (1000 900) 10
ENDTRACKS
BUSES 1
BUS B1
3
2
WIDTH 2
10
10
ENDWIDTH
BIT 0
L2 (0 795) (30 805)
L2 (970 595) (1000 605)
ENDBIT
BIT 1
L2 (0 745) (30 755)
L2 (970 695) (1000 705)
ENDBIT
BIT 2
L2 (0 645) (30 655)
L2 (970 745) (1000 755)
ENDBIT
ENDBUS
ENDBUSES
OBSTACLES 17
L1 (175 245) (185 255)
L1 (175 395) (185 405)
L1 (355 145) (365 155)
L1 (355 295) (365 305)
L1 (355 445) (365 455)
L1 (515 245) (515 255)
L1 (525 395) (525 405)
L1 (735 195) (925 505)
L2 (175 245) (185 255)
L2 (175 395) (185 405)
L2 (355 145) (365 155)
```

```
L2 (355 295) (365 305)
L2 (355 445) (365 455)
L2 (515 245) (515 255)
L2 (525 395) (525 405)
L2 (255 595) (450 900)
L2 (735 195) (925 505)
ENDOBSTACLES
```

The output in Figure 2 would be:

```
BUS B1
BIT 0
PATH 9
L2 (30 800) (220 800)
L1 (220 800)
L1 (220 200) (220 800)
L1 (220 200)
L2 (220 200) (620 200)
L1 (620 200)
L1 (620 200) (620 600)
L1 (620 600)
L2 (620 600) (970 600)
ENDPATH
ENDBIT
BIT 1
PATH 9
L2 (30 750) (140 750)
L1 (140 750)
L1 (140 100) (140 750)
L1 (140 100)
L2 (140 100) (660 100)
L1 (660 100)
L1 (660 100) (660 700)
L1 (660 700)
L2 (660 700) (970 700)
ENDPATH
ENDBIT
BIT 2
PATH 9
```

```
L2 (30 650) (100 650)
L1 (100 650)
L1 (100 50) (100 650)
L1 (100 50)
L2 (100 50) (700 50)
L1 (700 50)
L1 (700 50) (700 750)
L1 (700 750)
L2 (700 750) (970 750)
ENDPATH
ENDBIT
ENDBUS
```

The program should be named as "bus_router" and will be executed as:
`./bus_router <input file name> <output file name>`

VII. Test Cases and Evaluator

Two example test cases are released with the problem announcement in example.tar.gz. The first example case, example_1, is the case in Figure 2. The second example case, example_2, is a case with 5 buses with uniform track configuration. Note that example cases will **NOT** be used in the final grading.

Evaluator is released in the form of a tar file with the name of eval-<*version*>.tar.gz. The evaluator is written in python and compiled into an executable with shared libraries in a single directory. To run the evaluator, use the following command:

`./<version>/eval <input file name> <output file name>`

The evaluator checks the following:

- Input and output files format
- Route is successful or not
  - Connectivity
  - On track
  - Topology
- Spacing violation
- Calculate cost

Below is an example of the output of the evaluator of case example_1. If you experience any bug when using the evaluator, please let the topic chair know.

```
[INFO ] Parsing input file... done (0.02 seconds)
[INFO ] Parsing ouput file... done (0.01 seconds)
[INFO ] Initializing grader... done (0.00 seconds)
[INFO ] Checking bus B1
[INFO ]   Checking connectivity... done (0.00 seconds)
[INFO ]   Checking tracks... done (0.00 seconds)
[INFO ]   Checking topology... done (0.00 seconds)
[INFO ]   Cost Cw = 5 * 1.90 = 9.48
[INFO ]   Cost Cs = 1 * 1.67 = 1.67
[INFO ]   Cost Cc = 5 * 1.70 = 8.48
[INFO ] Cost CR = 19.62
[INFO ] Penalty Ps = 8 * 0.00 = 0.00
[INFO ] Penalty Pf = 200 * 0.00 = 0.00
[INFO ] Total cost C = CR + Ps + Pf = 19.62
```

Figure 20 Evaluator checking example_1 output

VIII. Reference

J. Cong, Jie Fang and Kei-Yong Khoo, "DUNE-a multilayer gridless routing system," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 5, pp. 633-647, May 2001.

X. Alpha Test Announce

	example_1			example_2			alpha_1		
Rank	Routing Cost (CR)	Pentalty Cost (CP)	Total Cost (C)	Routing Cost (CR)	Pentalty Cost (CP)	Total Cost (C)	Routing Cost (CR)	Pentalty Cost (CP)	Total Cost (C)
1	19.58	0	19.58	48.06	632	680.06	0	600	600
2	19.62	0	19.62	0	1000	1000	10.46	752	762.46
3	19.76	24	43.76	117.6	1376	1493.6	N/A	N/A	N/A
4	19.92	48	67.82	89.54	2560	2649.54	N/A	N/A	N/A
5	0	200	200	98.04	2688	2786.04	N/A	N/A	N/A

✧ Please make sure your binary can be executed on CIC server

- The g++ version on the server is 4.4.7 under /usr/bin/g++. If you uploaded binary compiled elsewhere, there's a chance C++ shared library linking could fail with error message such as "GLIBCXX_x.x.xx not found". This means that the binary is compiled with C++ library that is not compatible with the CIC server. The safest way is upload your source code and compile it on CIC server
- If you're using Makefile generated by third-party tools, e.g. Qt, be careful that it might depend on third-party binaries, e.g. qmake, that might not be available on CIC server
- If your binary require any third-party tools, the best solution is link your binary with third-party **static** libraries (.a extension). If the third-party tool only comes with a dynamic version (.so extension), and it is not located at server's default library locations, /usr/lib and /usr/lib64, please copy the shared library to the team directory. The evaluation script will add the team directory to its LD_LIBRARY_PATH environment variable.
- If you have any third-party tool request on CIC server, please let the organizer know. It would be better if you know the yum package name, e.g. yum install gnuplot. Also, if you need to compile the third-party tool along with your program, remember to request for the *-devel package.

✧ Please make sure your binary is named as "**bus_router**" as stated in the problem description. Other names, such as **team name**, **a.out**, **main**, etc., makes the automatic evaluation difficult

✧ Please use the released evaluator to check your results. If you found bugs or encounter problems when using the evaluator, please let the topic chair know.

XI

No Ranking announced.

✧ The final test testcases will consist these 5 beta test testcases plus 3 hidden testcases

✧ The final test testcases have similar design size and number of nets (around 1k~10k nets) to route

✧ The original runtime for beta testcases was set to 30min for each case. However, most of the contestants program will timeout under this setting. So, the runtime for each case is set to **3hrs**. In order to guarantee fair memory usage across evaluation of all contestants programs, the programs are evaluated serially. This takes huge amount of testing time. The goal of runtime for each testcase will be set to **1hr**. If your program could finish beta test testcases within 1hr, then the hidden case should be fine.

✧ The updated evaluation program (bug fixes) and beta test testcases are released along with the results.

XII

1. The final rank is determined by the “sum of all cost of all cases”. The smaller, the better
2. The 3 hidden cases used in final testing will not be released due to confidential reasons
3. The checker used for final testing use the following rules:
   a. Connectivity check for pin connection allow both centerline-touched or box-touched
   b. Via can produce spacing violation with wires/pins/blockages (as answered in previous Q&A) if the position of the via overlaps with the box of the other shape. For example, if a via is at (5 5), and there’s a horizontal wire (0 3) (10 3) w/ width of 3, then the two has violation if they belong to different nets
4. All testcases are allowed 1hr runtime. The size of the hidden cases are similar those of beta test
5. If you see your program crash/segmentation fault/abort on all cases, and the reason is due to third-party tools or environment settings, please let the event organizer know immediately. However, you cannot change the previously uploaded binary in any way.

XIII. FAQ

Q1. The spec told us that the tracks may not cover the whole design, so my question is: Can we make use of those regions which are not cover by the tracks or we can only put wires on the tracks to avoid routing failed?
A1. Regarding routing on- and off-track is explained in Paragraph 2 in Section V.A
"... For the second requirement, any wire off-track or has bus width constraint larger than the track width constraint is considered as route fail. ..."
In short, off-track routing is considered as route fail, and the penalty of route fail is explained in Section V.C.
The pins are guaranteed to overlap at least 1 legal track, but the total routable region may or may not have valid track solution for routing all the buses.
It is up to the contestant to perform trade-offs to minimize the overall cost.

Q2. In this problem, we have to route many buses, where each bus has many bits(a wire for a bit). For each input(test data), if there are more than one buses, is it true that we have to "route all the buses on the multilayer where any two bus can't collides" or to "route all the buses separately on different multilayer but with the same given constraints, which seems like a input(test data) with many small test data".?
A2. The routing requirement is more close to "route all the buses on the multilayer where any two bus can't collides" you mentioned. Though I'm not sure what you mean by "collides", do you mean short violations? The goal is to routing all buses on-track while optimizing routing cost, which includes spacing violation (spacing violation covers short violation). The buses can share the available routing layers if that's what you're asking, e.g. bus 1 routes on M0 and M1, and bus 2 can also route on M0 and M1

Q3. The statement says a wire can't be too close to other wires of different bits. Then is it able to be close to wires from other buses (or even overlap with them)?
A3. No. Wires on the same layer, belong to same bus or not, need to keep at least the distance of the spacing value of that layer from each other.

Q4. If so, it may occur that we use the same track for different buses. Then how is the length and width of the track defined? Does it mean we have to build it more than once (for each bus), or just once with the largest width?
A4. Answered in A3.

Q5. Is it that the size and shape of PINs have nothing to do with the width limitation of buses on each layer? Or can I say that the only usage of PINs is to point out one (or more) possible starting tracks of a bit (and when calculating the cost of compactness)?
A5. Correct, the size of the pins have nothing to do with the width limitation. I'm not quite sure of the second part of the question about the usage of the pins. In real design, the placement of pins and definition of tracks are carefully planned so that design rule violations in the routing stage are minimized. In routing specifically, I believe you could, like you said, just view them as starting and ending points to find tracks to route.

Q6. When there are multiple PINs of a bit, How is the topology checked and how is the compactness defined? Is it that all pairs of PINs are checked in the way mentioned and simply sum up the compactness?
A6. Let me use the right picture of Figure 14 to explain. The topology is considered as the same by satisfying the four criterions mentioned in Sec. V-A, namely, number of wires, layer sequencing, direction, and ordering. This should not be any different when dealing with 2-pin buses or multi-pin buses. The compactness cost of a bus is the sum of each segment's ratio of the width of the segment over the lower bound width of the segment as defined in Sec. V-B. In this figure, there are 3 segments, 2 horizontal ones (let's define them as seg1 and seg2) and 1 vertical one (let's define it as seg3). The compactness cost this bus is then (width of seg1 / lower bound width of seg1) + (width of seg2 / lower bound width of seg2) + (width of seg3 / lower bound width of seg3).

Q7. When will the test cases of problem B be released?
A7. The test cases are expected to be released in mid June.

Q8. Regarding "neighboring layers do NOT guarantee to have different routing directions", how to place vias between two neighboring layers with the same routing direction? Do such vias need to be on-track?
A8. Any point within the overlapping wire interval is valid. E.g. to connect M0 vertical wire (10, 10) (10, 50) and M1 vertical wire (10, 30) (10, 80), the via is valid in any y coordinate between [30, 50]

Q9. Regarding the spacing rule, is there any via-to-obstacle spacing rule or line-end-to-obstacle spacing rule?
A9. No additional spacing rules need to be considered. The rule of thumb is treat the wire as a rectangle as shown in Fig. 19. All sides need to keep at least the distance of

the spacing rule to other shapes. The focus of this problem is planning bus on tracks. Detailed DRC fixing will be handled by other engine.

Q10. Is it guaranteed to have at least one track CENTERLINE (the given track coordinate) or at least one track AREA (expanded rectangular area) to overlap with each pin?
A10. Yes. As described in the 6th bullet, Sec. III-E, "Each pin shape is guaranteed to overlap with at least one track which the width constraint of the track is larger than or equal to the bus's width constraint on that layer."

Q11. Does all the vias in a stacking vias need to be on-track? For example, if there is a track on L1 (0 10) (100 10) and a track on L3 (10 0) (10 100) but no trackpassing (10 10) on L2, can we use stacking vias L1 (10 10) L2 (1 0 10) to link L1 and L3 at (10 10)? -2018-06-19
A11. Stacking vias does not need to be on track

Q12. Can the vias in a stacking vias pass through an obstacle? For example, if there is a track on L1 (0 10) (100 10) and a track on L3 (10 0) (10 100) and an obstacle L2 (0 0) (20 20), can we use stacking vias L1 (10 10) L2 (10 10) to link L1 and L3 at (10 10)? -2018-06-19
A12. No. Via location point cannot overlap with obstacles or shapes (wires or vias) of other nets. If it does, it is counted as spacing (short) violation.

Q13. Can the vias in a stacking vias pass through another routing path? For example, if there is a track on L1 (0 10) (100 10), a track on L2 (10 0) (10 100) and a track on L3 (10 0) (10 100), can we use stacking vias L1 (10 10) L2 (10 10) to link L1 and L3 at (10 10) for one bus and a path L2 (10 0) (10 100) for another bus at the same time? -2018-06-19
A13. No. As stated above

Q14. If I rout L1 (20 30) (50 30) L1 (50 30) L2 (50 30) (50 40) for one bus and L1 (50 30) (70 30) L1 (50 30) L2 (50 5) (50 30) for another bus, well it be consider as routing fail due to short violation? -2018-06-19
A14. It is considered as spacing (short) violation, not route fail. As stated in Sec. V-A, a bus is considered as route success if the 3 requirements are met. In this case, the first bus has 1 spacing violation on L1 at (50 30) with the second bus, and 1 spacing violation on L2 at (50 30) with the second bus.

Q15. Are the physical locations of the same pin shape in different bits is guaranteed in the same x or ycoordinate?
For example, in fig.1 the left pin shape are in the same x coordinate, and fig.2 are not in the same x-coordinate

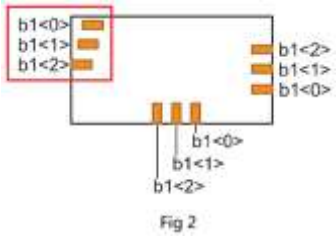


Fig 2

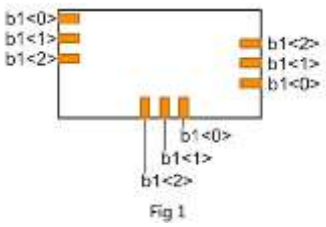


Fig 1

A15. You can assume they have either the same x (vertically distributed) or y (horizontally distributed)

Q16. For the input format, following the BIT <bit name> are the pin shape in this bit, are this pin shape order in different bits same? (in the following example, bit1's pin shape order are different)
Take fig.2 in document for example, Is it impossible the input format are
A16. The pin input order of one bit does not guarantee across all bits. Your program needs to be able to group pins from different bitstogether. The input only guarantees the order of the pins in one group should be the same or the reverse order in other groups.

Q17. As mentioned in the problem definition (page 7), the value of the width constraint is guaranteed to
be multiples of 2. However, we found that some tracks in example_1.inputhave width constraint 5, which contradicts the problem definition.

A17. The track with in the example is wrong. Updated in the new example tar file

Q18. We would like to clarify the definition of "On track". If a track (100,100) (100,200) with width constraint 20, is a wire
(110,100) (110,150) with width 2 on track? In other words, if a wire (with its width) is in the region of a track (with its width expanded), does it satisfy the definition of "On track"?

A18. No. The line (formed by the 2 end points) of the wire needs to overlap with the line of the track.

Q19. Will you release the evaluation program?

A19. The evaluation program will be released along with alpha test results.

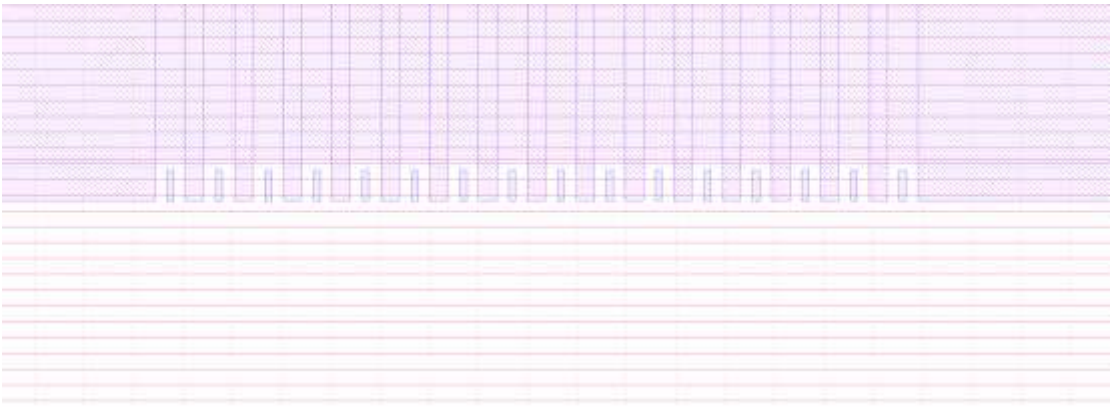
Q20. Does all the vias in a stacking vias need to be on-track? For example, if there is a track on L1 (0 10) (100 10) and a track on L3(10 0)
(10 100) but no track passing (10 10) on L2, can we use stacking vias L1 (10 10) L2 (10 10) to link L1 and L3 at (10 10)?
A20. Stacking vias does not need to be on track

Q21. Can the vias in a stacking vias pass through an obstacle? For example, if there is a track on L1 (0 10) (100 10) and a track on L3(10 0) (10 100) and an obstacle L2 (0 0) (20 20), can we use stacking vias L1 (10 10) L2 (10 10) to link L1 and L3 at (10 10)?
A21. No. Via location point cannot overlap with obstacles or shapes (wires or vias) of other nets. If it does, it is counted as spacing (short) violation.

Q22. Can the vias in a stacking vias pass through another routing path?
For example, if there is a track on L1 (0 10) (100 10), a trackon L2 (10 0) (10 100) and a track on L3 (10 0) (10 100), can we use stacking vias L1 (10 10) L2 (10 10) to link L1 and L3 at (10 10) for one bus and a path L2 (10 0) (10 100) for another bus at the same time?
A22. No. As stated above

Q23. If I rout L1 (20 30) (50 30) L1 (50 30) L2 (50 30) (50 40) for one bus and L1 (50 30) (70 30) L1 (50 30) L2 (50 5) (50 30) for another bus, well it be consider as routing fail due to short violation?
A23. It is considered as spacing (short) violation, not route fail. As stated in Sec. V-A, a bus is considered as route success if the 3 requirements are met. In this case, the first bus has 1 spacing violation on L1 at (50 30) with the second bus, and 1 spacing violation on L2 at (50 30) with the second bus.

Q24. In the first two bus of example_2,
The bit-pins of the bus on Layer_m3 is like the image below



The tracks in Layer_m3 are horizontal, so we should change the layer first;
However, in all the other Layer (m4~m7),
The position of these bit-pins are all covered by obstacles, just like the image below,
So we can't even change the layer for these pins.
A24. The testcase has been updated after alpha test and higher metal blockages have been removed

Q25. The second question is about layer-changing:
Is it valid for us to change from one layer to another layer which is not adjacent with current layer?
Or we can only change the layer to
the adjacent layer one by one until we reach our target layer?
A25. You have to change layers one-by-one. You can use stacking vias to achieve changing layers at the same point. E.g. L1 (10 10) and L2 (10 10) represent 2 vias from L1-L2 and L2-L3

Q26. As mentioned in the problem definition (page 7), the value of the width constraint is guaranteed to be multiples of 2. However, we found that some tracks in example_1.input have width constraint 5, which contradicts the problem definition.

A26. The testcase have been updated and the track with constraint is 6 now

Q27. We would like to clarify the definition of "On track". If a track (100,100) (100,200) with width constraint 20, is a wire (110,100) (110,150) with width 2 on

track? In other words, if a wire (with its width) is in the region of a track (with its width expanded), does it satisfy the definition of "On track"?

A26. The wire center has to be exactly on the same line of the track. In your example, a wire with width 2

Q27. Does it happen that different direction on neighboring layers?

A27. Yes. Neighboring layers can have different preferred routing directions

Q28. First, for example_1, we find that some spacing violations between wires and the boundary seem unavoidable. To be more specific, the design boundary is "(0 0) (1000 1000)" with a spacing of 30 required on L2. For example, the first pin of bit 0 is "L2 (0 795) (20 805)". Therefore, if a wire wants to touch the pin, its lower x should be at least 20. However, its spacing to the boundary 0 will be 20 and less than 30. Is it a benchmark issue? On the problem statement pdf, the first pin of bit 0 becomes "L2 (0 795) (30 805)", which looks OK.

A28. Fixed in the new example file released after alpha test

Q29. for example_2, some spacing violations between wires and blockages also seem unavoidable. The detailed information is as follows. Essentially, the pin is on m3 but needs to go to m4 due to the track direction (there is no m2). But on m4, there is an obstacle fully cover the pin.

Layers:
  m3 horizontal 90
  m4 vertical 90
The second pin of bit net22<0>: m3 (112765 82660) (112835 82970)
The corresponding pin of the track: m4 (112800 0) (112800 139285) 70
A neighboring obstacle conflicted: m4 (100830 82660) (118590 100635)

A29. Same as above

Q30. we see many wire-via spacing violations reported by the evaluator. But in the problem statement, it is said that each wire should keep at least the distance of the spacing constraint of the layer with the following objects: 1. other wires of different bits, 2. obstacles, 3. designboundary. Is the wire-via spacing violation really counted? If yes, how does it calculated?

A30. The via-wire or via-obstacle violations are only counted when they overlap with each other (short). E.g. if you have a wire (10 10) (10 100) with width of 10, then the wire box is (0 10) (20 100). Placing any different-net vias within this box is considered as violations.

Q31. I have downloaded the new examples of Problem B from the contest website. And may I ask one more question about the contest problem?
As shown in Figure 1, the blue rectangle represents the pin shape, and the red rectangle represents the wire shape. I was wondering if the pin and wire in Figure 1(a) and Figure 1(b) are connected successfully or not?

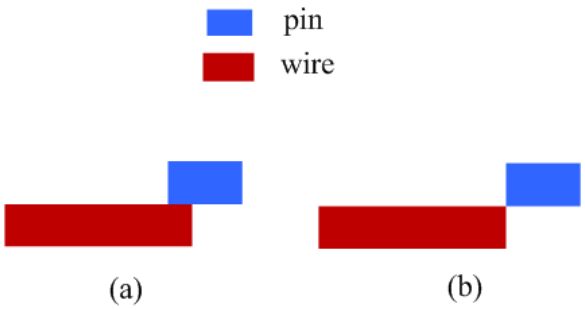


Figure 1

A31. The checker checks for wire boundary and pin boundary overlap. So, if the edges touch each other, they are considered as connected. Though the center line of the wire touching the pin is preferred (since in detailed routing there will be less DRCs), currently there is no penalty on these type of connections

Q32. For different bits of a bus, can they have misaligned via stacks? To be more specific,
suppose there are bit1 and bit2, and layers L1 and L3 are horizontal. Can bit1 come from
L1 to L3 earlier than bit2? For example, is the following routes legal (in terms of
topology consistency):

bit1:
L1 (0, 0) (10, 0)
L1 (10, 0)
L2 (10, 0)
L3 (10, 0) (30, 0)

bit2:
L1 (0, 10) (20, 10)
L1 (20, 10)
L2 (20, 10)
L3 (20, 10) (30, 10)

A32. The topology consistency only cares about wire ordering. In your example, since there are only 2 bits, the wire ordering on L3 can be the reverse ordering of that on L1 as stated in the document. In short, via stacks play no role in topology consistency checking, only wires do.
"

Q33. In alpha_1.input,
the second bit pin of bus1 and bus2 is not overlapped with any track,
Just as the image below.

A33. Updated case to add tracks on m9

Q34. I have a question about CAD contest problem B.
In the FQA part (A30), you say "The via-wire or via-
obstacle violations are only counted when they overlap with each other (short).
E.g. if you have a wire (10 10) (10 100) with width of 10, then the wire box is (0 10)
(20 100). Placing any different-net vias within this box is considered as violations.",
but the evaluation program told me "Spacing violation (need >=90) between WIRE A<0>
m6 (12625 12800) (12695 139195) and VIAA<1> m6 (12780 94720) (12780 94720)", so
which one is correct?

A34. It is the evaluation programs bug. It will be fixed.

Q35. Can the output file be evaluated even if the program is interrupted abnormally?

A35. As long as the output file are written out. You could make your program to write out solutions during execution

Q36. And does the running time of the program have any effect on the final ranking?
I am so sorry for disturbing your busy work, and thanks for your great help and your hard work.

A36. No. The runtime only serves as a threshold to cutoff program.

Q37. In Figure 19 (in the problem description), the wire P_2 and
the obstacle O_1 do not have spacing violation.
Is it correct? Or, the end of line spacing is not considered in this contest?

A37. The figure is misleading, p2 and O1 HAS spacing violations

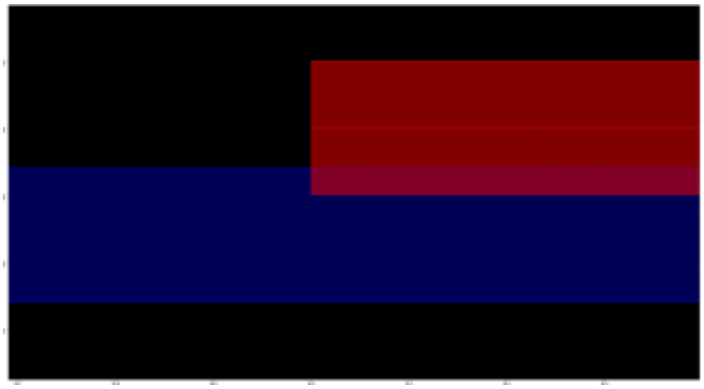
Q38. We found two issues about bus1 and bus2 in alpha_1.
First, there will always be spacing violation in the right-hand side of bus1 and bus2 if bus1 and bus2 are ever routed.
Second, there may be a bug in the evaluation program, since it only found the spacing violation of bus1, but not bus2.
We have attached our result of alpha_1 and the result of evaluation in this email.

A38. Thanks. The bugs will be fixed in the beta test results

Q39.In "alpha_1.input" and "example_2", we found that some bit groups are not reachable without spacing violations (e.g., bus1 and bus2 of alpha_1; bus0 and bus1 of example2). We suggest that every bus must have at least one violation-free pin access (with the same topology). Otherwise, it may be not reasonable that the output with no paths gets a better score than that with no open nets.

A39. In real case, there's no guarantee that every pin has a violation-clean connection. Though there will be higher penalty to the route fail results

Q40.



As shown, the blue wire and the red wire are not in the same layer, but it seems to be regarded as connected in the evaluator if the via is inserted at the overlapped region. Thus, Is the via between two wires or a wire and a pin not necessary to locate at the center of the wire?

A41. It is a bug of the checker. According to description, the via should be located at the center of the wire to be considered connected. The current checker checks only whether the via lands inside of the expanded wire shape.

Q42.Is there any memory limit?

A42. The hard constraint is the memory of the CIC server. For problem B, each contestant's program is run serially so that each one could have full memory usage of the server. Note that there will be some background processes that consume memory as well.

Q43. Is there any new benchmarks that will be released before the deadline?

A43. Yes. And the final test binary upload deadline of problem B is postponed

Q44. The bit-
shapes from bus3_bit156 to bus_bit160 in beta_5.input doesn't follow shape ordering. Should it be treated as unroutable or it is a mistake of benchmark?
A44. It will be fixed

Q45. Neighboring layers do NOT guarantee to have different routing directions.
If the wire of a bus goes on the same direction but in neighboring layer,
is it routed fail due to topology or any violations ?
ex. BIT0 L2 (10 10) (10 100)
BIT1 L3 (20 20) (20 100)
BIT2 L2 (30 30) (30 100)

BTW, Since that neighboring layers do NOT guarantee to have different routing directions,
I was wondering that what is this case look like and the detail of evaluation methodology in this kind of case.

If this kind of case is not released, it is really hard to get a good solution.

A45. It is considered as route fail due to different topology

Q46. In the Q&A A41, "the via should be located at the center of the wire to be considered connected."
So if a case that the pin(red) has misaligned track(blue)
on different layer as figure in Q40, then we still need to put the via at the center of the wire ?
If so, the via probably does not connect the pin, is it right ?

A46. In beta and final tests, all pins should have overlap tracks, though not necessary at the center of the pin
So, there should always exist a point for the via where is overlaps with the pin, and at the same time, overlaps with the track (center of the wire)

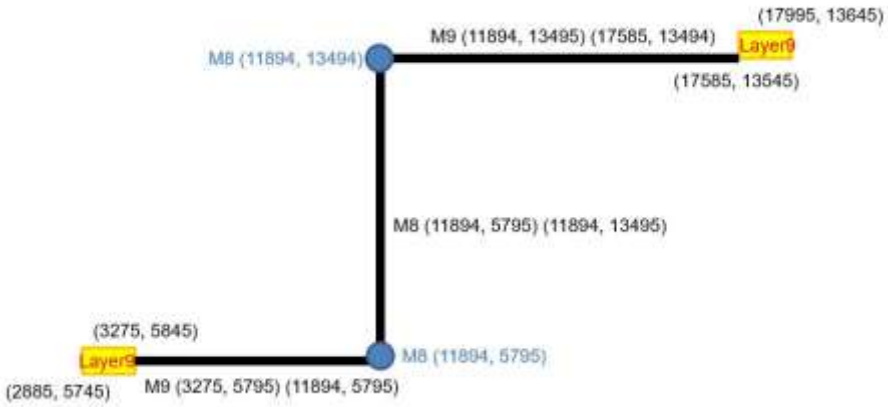
Q47. The latest version of checker doesn't work like Q&A description.
Specifically, Q&A 31 says wires and pin shapes are connected if those edges touch each other.
However, the latest version of checker treat two buses in "alpha_1.input" with pin shape misalignment as routing fail.
While the previous version of checker treat them as successful routes.
Which rule should we follow?

A47. Please use the newest checker as the reference.
For beta and final test testcases, all pin shapes will have tracks overlap with them

Q48. For example in alpha 1, fig 1 is the input file and the output file of bus 2.
Fig 2 shows the result from our output file.
We think it might be an available solution and we did evaluation with version 4 evaluation program.
The result succeeded, but when we use the newest version of the evaluation program, all the bus were consider as opened.
We are confusing whether it is we having a wrong output or is the evaluation program problem.

```
BUSES 3
BUS bus1
16
2
WIDTH 3
70
110
100
ENDWIDTH
BIT C<1>
m9 (2885 5745) (3275 5845)
m9 (17585 13545) (17975 13645)
ENDBIT
```

```
BUS bus1
BIT C<1>
PATH 3
m8 (11894 13495)
m9 (11894 13495) (17585 13495)
m8 (11894 5795)
m8 (11894 13495)
m8 (11894 5795) (11894 13495)
m8 (11894 5795)
m9 (3275 5795) (11894 5795)
ENDPATH
ENDBIT
```



A48. The connectivity checking is now using the “line” of the wire to check whether it connects to other shapes

In the alpha testcase, no tracks overlaps with some of the pins, so no wires could connect to the pins

Please focus on the beta test testcases for more realistic scenario