Multilayer Standard Cell Detailed Router

Routing Group 10: Namseok Kim and Mohammad Mohammad

I. RESULT

A. *Successfully Routed Cells*

Table I shows the results for cells that passed DRC.

TABLE 1. RESULTS FOR MINIMUM SPACING RULE

Metal width: 65, metal spacing: 55, via/contact ext: 10, metal area: 7800, via size: 65x65							
Cell	*LVS*	*DRC*	*minimum area rule*	*Runtime*	*number of vias*	*metal1 length(nm)*	*metal2 length(nm)*
INV_X1	1	1	1	0.28s	0	1805	0
LOGIC0_X1	1	1	1	0.28s	0	786	0
AND2_X1	1	1	1	0.5s	4	2909	800
AND2_X4	1	1	1	0.81s	10	6065	2325
INV_X32	1	1	1	0.33s	24	16891	11935
DFF_X2	1	1	0	5.46s	22	12369	6800

B. *Design Rule Explorations*

As shown in Fig. 1, the minimum spacing rule is more restricted than the minimum contact/via extension rule. The Routable range of minimum spacing was 55~65nm, while the routable range of contact/via extension was 10~35nm. Such result is due to the fact that the minimum spacing is applied on both vertical and horizontal directions at the same time while the extension rule is applied only to one direction, either vertical or horizontal. Therefore, the minimum spacing rule has less flexibility and is more critical for routability.

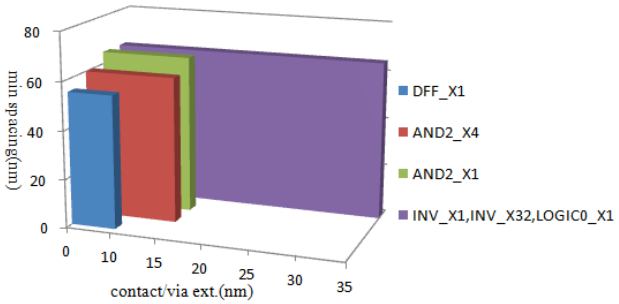

Fig. 1. Routable combinations of minimum metal spacing rule vs. contact/via extension rule for each cell. It was checked at every 5nm increment.

II. ROUTING APPROACH

A. *Net Ordering*

The router routes nets sequentially. Therefore, net ordering is crucial since a routing solution for one net impacts the routing of other nets [1]. We prioritize nets based on the availability of routing alternatives. Reference [1] suggests that a practical prioritization would be routing nets with less routing alternatives before other nets. For the given test cases, we determined that nets with smaller HPWL have limited routing choices due to design rule restrictions, and thereby, have less routing alternatives. The router therefore routes nets in an ascending order based on HPWL.

B. *Grid and Blockage Assignment*

We use two layers of fine grids (5nm x 5nm) to minimize offset and maximize resolution. When routing two points, we bound the breadth search to a framed grid to reduce runtime and memory consumption. Metal1 and contact blockages are appended to layer0 of the grid, and metal2 blockages are appended to layer1. For a multi-pin nets, objects of the current net being routed are not added as blockages, which allows overlaps.

An example of a blockage appended to a grid is shown in Fig. 2(b). An offset margin is given to the blockage on the grid, which is the blue boundary. Spacing requirement, metal width/2 + minimum spacing, is added to the blockage, resulting in the red boundary.

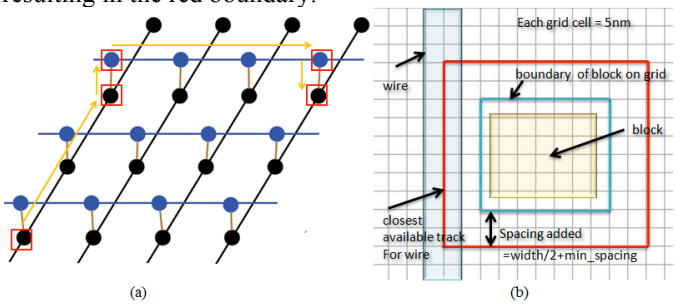

Fig. 2. (a) The yellow lines are a path of modified A* algorithm. The red boxes are a compressed set of path. (b) A blockage added on the grid.

C. *Modified A* Algorithm*

A* algorithm is modified to have a z-axis in addition to x and y-axis, and searches in a 3D grid graph. It guarantees a path with minimum via cost. The cost function has the same form as the original A* algorithm, f = g + h [2]. The only modification is the added vertical distance in Manhattan distance computation, and a path between different z values has a higher weight. Each node has three children nodes.

D. *Path Compression*

The path returned from A* algorithm contains only the source, target, and the via crossing grid points as shown in Fig. 2(a). Such simplified path is then passed to the trialConnect and offset correction functions.

1) *trialConnect function*

If a compressed path set has only two points (i.e. only the source and the target), a vertical path between (x_0, y_1, z_0) and (x_0, y_2, z_0) is the only possibility. If the set has four points with z=0 for both source and target, it's a horizontal path with $(x_1, y_0, 0)$, $(x_1, y_0, 1)$, $(x_2, y_0, 1)$, and $(x_2, y_0, 0)$. The trialConnect function checks if there is a blockage between two points by checking the number of points in the compressed path set.

2) Offset Correction

Let $\{(x_s,y_s),(x_t,y_t)\}$ represent the original technology coordinates for two points to be routed (source and target). And let $\{(x_s',y_s'),\ldots..,(x_t',y_t')\}$ be the path set from the source point to the target point. The apostrophe means that the points are fitted to the grid layers and hence, may not be correctly aligned with the original coordinates. The points in the latter set have offset up to 4.9nm due to discrete grid size. Hence, it is necessary to correct the offset to align wires with contacts.

The offset between (x_s,y_s) and (x_s',y_s') is applied to points close to the source, and the offset between (x_t,y_t) and (x_t',y_t') is used to correct points close to the target. Because the offset at the source and the target are slightly different, the correction at midpoints could have some error as the distance between source and target increases. However, the error is very small (<< 5nm) and design rules are still satisfied because of the conservative margin assignment of blockages on the grid as shown in Fig. 2(b).

E. Multi-pin Connection Algorithm

For a net that contains more than two points, each two points are routed sequentially. Near-optimality is obtained through a points ordering algorithm that enables formation of steiner points and a 3-bend net. The algorithm does not cause a design rule violation due to overlaps.

Let P and Q be sets of points where both sets belong to the same net. Let P represent a set of points that are already routed and Q represent a set of points that are to be routed.

Several rules are presented that determines the ordering of points in a net.

1) Priority rule

From the first point A in set Q, search if the set Q has any point which has the same x coordinate as A. If any, connect A with the closest point B among these points. Delete A and B from set Q, append A and B to set P. Then, from the last appended point B in set P, search for every point in set Q which has the same x as B. Then connect B from the closest point C among these points. Delete C from set Q, and add it to set P. This rule is illustrated in Fig. 3(a).

2) Basic rule

The basic rule is applied when there is no more case for the priority rule to apply. This rule is illustrated in Fig. 3(b).

From the first point A in set Q, search for the closest point B in set Q. Connect A and B. Delete both from set Q and add them to set P. From the center of set P, find closest point C in set Q. From C, find closest point B in set P and connect C to that point. Delete C from set Q and add it to set P.

F. Correction of Design Rule Issues

1) Deter rule

This rule covers two major cases of design rule violations during a multi-pin connection. First case is when there is a blockage in between two vertically aligned contacts as shown in Fig. 3(c). The router checks for blockages between every vertical path by using trialConnect. If there is a blockage, it replaces B with next closest point C in set Q. Second case is when two contacts are slightly misaligned such that |x(contact1) – x(contact2)| < minimum spacing + minimum width. In this case, a deterring path is created as well.

2) Contact misalignment

This case is similar to the second case of deter rule, except that it is for nets with two points only. Blockages in between two misaligned contacts are checked in both directions as demonstrated in Fig. 3(d). If it's clear, both contacts are covered with one piece of metal. Otherwise, the avoidance of design rule violation is not guaranteed.

3) L-bend

The L-bend issue is shown in Fig. 3(e), in which the blue path creates a short bend which violates design rule. Assuming n^{th} point is the targeted contact in the compressed path set, the violation is detected when n-2^{th}, n-3^{th}, n-4^{th}, and n-5^{th} points have the same y values and the x difference between n-1^{th} and n-2^{th} points is smaller than required spacing. If the violation is caused by blockage A and B being too close to each other, only solution is eliminating either blockage A or B. However, if there is some space between blockage A and B as in Fig. 3(e), the violation problem is fixed by running our algorithm in reverse direction. As the red path describes it, the algorithm makes the first bend near the blockage A.

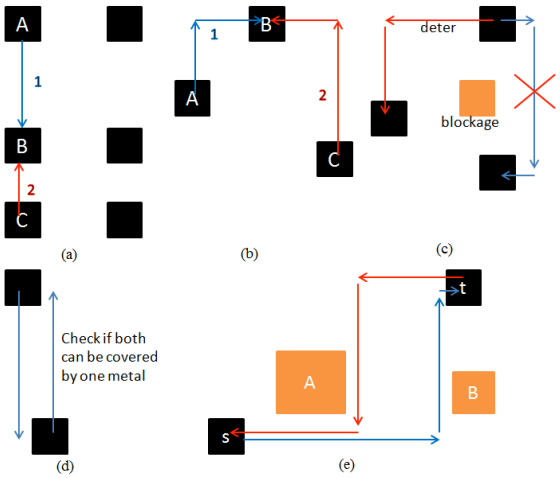

Fig. 3. (a) Priority rule. Arrows are connection directions and numbers indicate connection orders. (b) Basic rule. (c) The first case of deter rule. (d) Contact misalignment fix of a two pin net. (e) L-bend violation. The blue path occurs in the presence of blockage B. A solution is indicated by the red path.

G. Minimum Metal Area Correction Function

After routing of all nets is done, this function iterates through all metal shapes and finds violating shapes. Using trialConnect, blockages for two vertical directions are checked if there is enough space for the shape to expand. If so, an expanding shape is drawn.

REFERENCES

[1] C.J. Alpert, D.P. Mehta and S.S. Sapatnekar, *Handbook of algorithms for physical design automation*, CRC Press, 2008.

[2] P.E. Hart, N.J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths", *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, 1968, p. 100-107.

[3] Y. Xu, Y. Zhang and C. Chu, "FastRoute 4.0: global router with efficient via minimization", *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, 2009, p. 576-581.