# Octilinear Redistributive Routing in Bump Arrays

Renshen Wang
University of California, San Diego
La Jolla, CA 92093-0114
rewang@cs.ucsd.edu

Chung-Kuan Cheng
University of California, San Diego
La Jolla, CA 92093-0114
ckcheng@ucsd.edu

## ABSTRACT
This paper proposes a scheme for automatic re-distribution layer (RDL) routing, which is used in chip-package connections. Traditional RDL routing designs are mostly performed manually because the wire geometries are more flexible and therefore more difficult to handle on RDL than on chip. For example, octilinear routing is manufacturable in RDL and is widely adopted due to its higher efficiency than Manhattan routing. In this paper we devise a polynomial time octilinear RDL routing algorithm based on a grid network embedded in the bump array. The grid network is constructed to fully utilize the routing space as well as avoid any spacing violation. Detailed routing solution can be obtained following the min-cost max-flow in the network. Experimental results show the effectiveness of our router.

## Categories and Subject Descriptors
J.6 [**Computer Applications**]: Computer-aided design

## General Terms: Algorithms, Design

## Keywords
Interchangeability, 8-geometry, routing grid, network flow

## 1. INTRODUCTION
Flip-chip package is widely used due to its high performance and high density chip-board connections provided through an area bump array. With the increasing complexity of VLSI circuits and the increasing number of connections, routing between bumps and wire-bonding pads becomes a challenge. In recent IC designs, the I/O pads are placed in uneven distributions, so we need a Re-Distribution Layer (RDL) to connect these pads to the bump array. From package to board, a similar redistribution is also needed for the signal array to escape outward to other devices.

Compared with on-chip routing, RDL routing has the freedom of interchanging the destinations of the pads which makes it easier, but also has the difficulty of placing wires with different directions in a single layer. The redistribution layer (RDL) is usually a metal layer with 8-geometry wires, i.e. there are 45° wires together with horizontal and vertical wires on the same layer. As in Figure 1, the chip-level pin pads are connected with the package-level bumps through this layer.

We propose an algorithm to automatically route the wires in the redistribution layer (RDL), and thus hopefully enhance the productivity of the design process. Previously, some similar problems on interchangeable routing were introduced in [1] – [3]. In [1], a general RDL routing problem is proved to be NP-complete, and a heuristic network flow router based on Delaunay triangulation is used. Since the number of possible triangulations is exponential, no optimality is guaranteed for the algorithm. An RDL router by integer linear programming (ILP) on a routing graph is proposed in [2], where the routing edges are picked under a series of linear constraints. A network flow algorithm is adopted in [3] to route Manhattan wires on a routing graph. The problem with [2] and [3] is the inaccuracy of routing capacities with 45° wires. Figure 2 shows a "tile" with different capacities. The tile accommodates 4 vertical wires going through, or 7 wires through the diagonal line with 45° segments. So $t_m = 4$ is underestimating the total routing capacity and $t_m = 7$ is overestimating. More complex graph components may work for the wire capacities in a tile, but for incomplete tiles with missing pads, there are still no simple and effective graph models. The discrepancy between the graph capacity and the real geometric capacity is a main problem not resolved in previous works.


*GLIVLSI'09,* May 10–12, 2009, Boston, Massachusetts, USA.

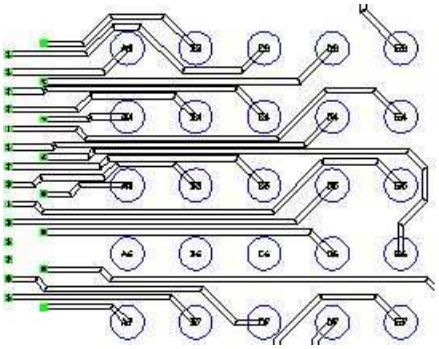



**Figure 1. Re-distribution layer (RDL) routing example**

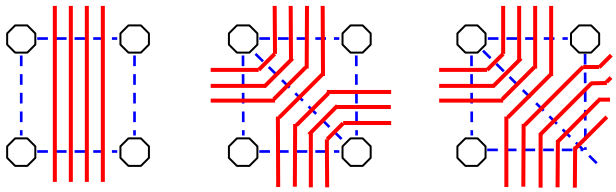



**Figure 2. "Tile" capacity differs with wire styles and bumps**

Our approach looks for an automatic RDL router applicable in general bump arrays, because complicated circuit designs nowadays may require bump arrays with non-regular shapes. Moreover, 45° wires are desired in redistribution layer routing in order to reduce wire length and improve routability. We devise an octilinear (8-geometry) RDL router based on a grid network to meet these requirements.. The grid network is constructed by horizontal and

vertical routing tracks and added with 45° tracks, which can guarantee both sufficient and necessary conditions of RDL routing with min-cost max-flow solutions. The wire density achievable in the grid is equal or close to the maximum density allowed by wire width and spacing so that the routability is well preserved in the grid, which is realized by a "line shift" technique. Detailed network construction is described in section 4. Experimental results in section 6 shows the effectiveness of the router on industrial and artificial RDL routing cases.

## 2. PROBLEM FORMULATION

In this approach, we assume the RDL routing is in a single layer. We have a bump array with n lines by m columns and a group of wire-bonding pads placed among the bumps. The objective is to assign one bump to each pad and connect them by a wire of width w, under the spacing constraint: spacing between a wire and a bump, pad or another wire is at least s. The parameters w and s are given by user.

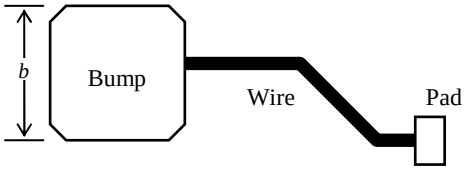



**Figure 3. Polygons of bumps, wires and pads**

The bump array may have different pitches between lines or columns, and may not be a full $n \times m$ array. Each bump pad is basically a square, or an octogon with four small corners chopped of a square. The wire-bonding pads are small rectangles, usually placed in rows over the chip. Figure 3 shows the polygons of the three types of objects on the redistribution layer.

Wires consist of wire segments in directions of horizontal, vertical or 45°. The connections between each pad and each bump are not specified, so we have the freedom to choose the connections. All wire-crossings can be avoided since we can interchange the destinations of any two wires and thus resolve every possible crossing.

While the 45° wire segments increases the routing difficulty because of variable routing capacities in the bump array (Figure 2), the interchangeability in the routing source and destinations helps us to reduce the problem to a flow-on-grid problem, as described in next section.

## 3. GRID NETWORK AND MIN-COST MAX-FLOW

A grid network can be used to model the routing capacity in the space among bumps. We first demonstrate this model in relatively easy Manhattan routing which only uses horizontal and vertical wire segments (4-geometry).

### 3.1 Manhattan (rectilinear) Grid

Assume the bumps in the array are square with the same size, and the rows and columns of the array are all aligned, with exact space for integer number of wires in routing channels. In an ideal array like this, optimal solution is guaranteed with a grid network model.

As in the array shown in figure 3, we construct a grid by adding routing tracks in the channels between bumps, which is called a *routing grid*:

1) For each horizontal channel or vertical channel $i$, the routing capacity $C_i = (\text{Width}_i - s)/(w+s)$. We put $C_i$ tracks in the middle of channel $i$ with spacing $(w+s)$.

2) On each channel segment with length b, place B tracks across the channel segment where $B = \lfloor (b+s)/(w+s) \rfloor$, also with spacing $(w+s)$ and at the middle of the segment.

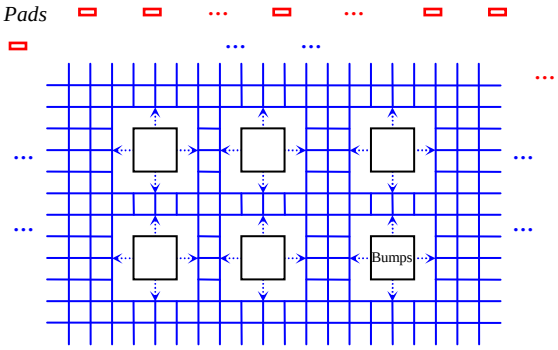



**Figure 4. Manhattan routing grid in a bump array**

Grid nodes are formed by crossing points of horizontal and vertical tracks, and this grid network can be regarded as an undirected graph (except some directed edges from bumps to grid nodes). If we only use Manhattan (4-geometry) wires, this graph with some extra directed edges from bump to grid (Figure 4) is sufficient to do basic RDL routing. Let each bump to be a source and each pad to be a sink, and the capacity of each node is one, the max-flow in the network is exactly the routing solution.

***Theorem 1***: In the bump array and routing grid, if each pad is placed on a grid node, a Manhattan RDL routing solutions exists *if and only if* the max-flow in the network reaches the number of connections required.

*Proof*: For Manhattan RDL routing and Manhattan grid, the proof is straightforward.

←: When there is a network flow solution in the grid, we simply put each wire along a unit of flow from bump to pad. Since each node has only unit capacity, any two units of flow can never converge at a node, and the distance between them is at least one unit of the grid. Thus the network flow induced RDL routing satisfies the spacing requirement.

→: When there is an RDL routing solution, we can shift each wire segment onto a grid line. So each wire becomes a unit of flow. Since the number of grid lines between two adjacent bumps is the maximum wire capacity Ci according to the spacing constraint, this shifting is always doable. So a routing solution also implies a network flow solution. □

Although the Manhattan RDL routing is a largely simplified version of our problem, this theorem justifies the methodology of using network flow based on a grid structure. Also it provides the basis of our following grid network constructions.

### 3.2 Octilinear Grid

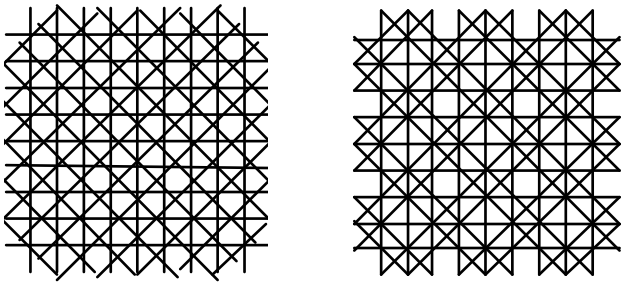
For practical octilinear RDL routing, we need to add additional 45° wire tracks in the grid network to enable routing in all the 8 directions. The most straightforward way is to duplicate the same grid, rotate it by 45°, and superimpose it on the Manhattan grid as in

figure 5(a). But this is not a valid construction of routing network, because too many crossing points are produced by the two Manhattan grid nets, and a large number of spacing violations exist between the crossing points.

The 45° tracks should be added with two constraints in order to be usable as the Manhattan grid in previous subsection:

1) No spacing violations in the network flow solution followed by the RDL routing solution.

2) No loss of possible routing solutions, i.e. If there is an RDL routing solution, there should be a corresponding solution by network flow in the graph.



**(a) Original wire tracks** **(b) After topological "line shift"**

**Figure 5. Octilinear grid network**

We use a "line shift" technique, which is shifting the 45° tracks as in figure 5(b). With the "line shift", the tracks are shifted onto the closest Manhattan grid nodes and excessive crossing points are eliminated. Only a few off-grid crossing points are produced by crossing 45° lines in the "X" shape of figure 5(b). This octilinear network structure is the key of our solution to RDL routing.

The shift of each line here is only topological, that is, the actual positions of these 45° wire tracks can stay as in the original grid of figure 5(a). The effect of "line shift" is only topologically attaching the lines to the grid nodes in order to eliminate spacing violations between crossing points.

The benefit of this "line shift" technique is a one-to-one correspondence of routing and network flow solutions similar to theorem 1, although with a more strict constraint.

***Theorem 2***: In the bump array and octilinear routing grid, if we guarantee that all the source and sink pads are properly located without spacing violations with wires, an octilinear RDL routing solution exists *if and only if* the max-flow in the network reaches the number of connections required.

*Proof*: With the assumption of no violations between source/sink pads and wires, we discuss the routing solutions.

←: When there is a network flow solution in the network, the RDL routing solution constructed by the flow segments is free of spacing violation.

First, no spacing violation exists between parallel flow units, since the wire tracks are placed with enough spacing. Though some 45° wire tracks look too close in the graph due to shifting, the actual spacing is still guaranteed.

Second, potential violations between grid node and 45° tracks, or between off-grid node and Manhattan track are all avoided. Figure 6 shows the two possible violations. On the left side, the space is too small between a flow unit on 45° track and the turning corner of another flow unit. However, if we take the min-cost max-flow where the cost is path length, the flow unit on the corner will not really pass the "turning point" close to the 45° track, because it must take the shortcut which is on another 45° track to achieve min-cost. Thus the two flow units will have a minimum distance no less than two parallel wire tracks.

The case on the right side of figure 6 is similar: there is an off-grid node too close to a horizontal wire track. But when there is a flow unit in the horizontal wire track, the off-grid node will not be passed by a flow unit as a "turning point", otherwise the flow is not min-cost. So the spacing here is also adequate.

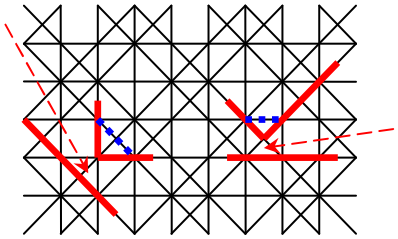


**Figure 6. Possible violations avoided by min-cost max-flow**

→: When there is an RDL routing solution, the network flow solution can be constructed by shifting the wires onto tracks. Since the tracks are originally placed as in figure 5(a), the number of tracks in any area is also the maximum number of wires possible, so we can always find a track for each segment of flow, with some possible "line shifts" to fit in the network of figure 5(b). □

The conclusion in this theorem holds with a constraint: when the sources and sinks, which are the bumps and pads to be connected, are placed with enough spacing between them and have no potential spacing violations with wires. This constraint can usually be satisfied in common cases because of the regular shapes of bump arrays and pad arrays.

The details of network construction within bump arrays and final RDL routing are described in following sections.

# 4. GRID NETWORK CONSTRUCTION

Based on the octilinear grid, RDL routing can be performed with a standard min-cost max-flow algorithm. In real RDL cases with flip-chip bump array, there are bumps acting as sources and obstacles. We cannot directly apply the network in figure 5(b). However, using the idea with topological "line shift", we can implement the methodology with more detailed construction of the routing network.

Considering the blocking obstacles, the network must be embedded in the bump array and need to avoid spacing violation between routed wires and bumps. We dissect the construction of network into different areas as follows.

## 4.1 Octilinear Routing Grid in Full Arrays

The locations of the Manhattan routing channels can be calculated from the bump coordinates. The routing capacity in a horizontal or vertical channel is $C_{ch} = \lfloor (Width_{ch} - s)/(w+s) \rfloor$, so we put $C_{ch}$ tracks in the middle of the channel, with minimum spacing $(w+s)$ between adjacent tracks. Besides, like in figure 3, we put $B = \lfloor (b+s)/(w+s) \rfloor$ cross-channel tracks orthogonal to the channel. Figure 7 is part of a network.

For 45° tracks, due to the shape of obstacles (bumps), the construction need to be decomposed into two parts, channel crossing areas and channel segment areas between crossings areas, as shown in figure 7.

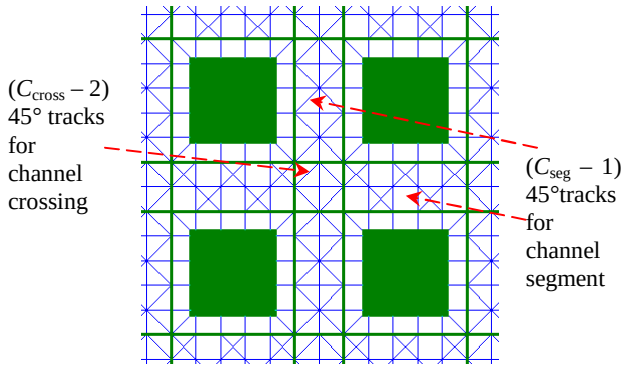



**Figure 7. Network construction in bump arrays**

1) Consider the crossing areas of horizontal and vertical channels in the bump array (the 2×2 grid in figure 7). The number of 45° tracks here is (Ccross – 2) in each direction, where Ccross is the capacity of 45° diagonal channel:

$$C_{cross} = \left\lfloor \left( (W_H + W_V)/\sqrt{2} - s \right) / (w+s) \right\rfloor$$

$W_H$ and $W_V$ are the widths of the horizontal and vertical channels. The condition here is slightly different with figure 5 and 6 because of the 4 bumps at the crossing. The routing capacity between two diagonally adjacent bumps is 4, but we only add 2 tracks in the central 2×2 grid. Since two more flow units can go through the turnings beside the bump corners, the diagonal capacity of the crossing area is correct. The real locations of 45° tracks are distributed at the center of the crossing area. We set the distance between adjacent tracks as the minimum grid unit ($w$+$s$) for the best utilization of space. For the topological connections of each 45° track in the network, we pick the nearest on-grid nodes along the track and shift the track on these nodes.

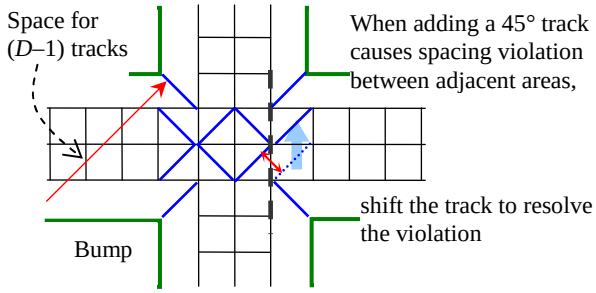



**Figure 8. Adding 45° tracks and spacing check**

2) For the channel *segment areas* between adjacent bumps and also between the crossing areas, 45° tracks in the segments enable wires to shift from one Manhattan track to another. The diagonal capacity $C_{seg}$ can be computed in the same way as $C_{cross}$.

$$C_{seg} = \left\lfloor \left( (Length_{seg} + Width_{seg})/\sqrt{2} - s \right) / (w+s) \right\rfloor$$

We put ($C_{seg}$ – 1) 45° tracks in each direction, so we can guarantee enough space for the out-coming wires from the bump corners (figure 8). The real locations of 45° tracks are also distributed at the center of the area and the topologically connect to the nearest on-grid nodes. But here we also need to check the spacing between the parallel 45° tracks on the boundaries of adjacent areas. If there is a spacing violation, we shift the tracks with inadequate spacing onto the same grid node as in figure 8 (right half), so that they are topologically connected and the violation is resolved.

By theorem 2 we know if pads are on grid nodes with enough spacing, an RDL routing solution can be constructed from a min-cost max-flow solution in the routing grid.

## 4.2 Octilinear Routing Grid in General Arrays

In practical flip-chip packages, we will have cases with bump array in non-rectangular shapes. A new type of routing area comes from the missing bumps in the bump array.

Basically we still use the idea in figure 5, but the exact optimal solution is not guaranteed. Because with missing bumps (empty areas), some channels with extra space for fractional capacity units maybe combined together. Unlike in a full bump array, the combined fractions can produce additional routing capacity in some non-regular shaped bump arrays, which is lost in our routing grid since all the Manhattan grid lines are aligned. However, the loss of fraction is less than 1 for each channel and less than $n$+$m$ in an $n$×$m$ array. So we still have an efficient approximation in terms of the number of bump-pad pairs routed.

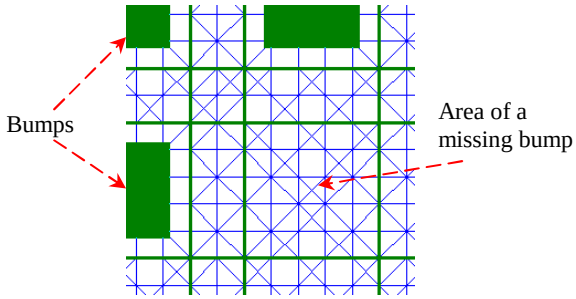



**Figure 9. Network construction at a missing bump**

The grid network at the missing bump area can be constructed by extending the tracks in the channel segments beside it (figure 9). Moreover, we add two diagonal tracks in the square area, since the constructions of 45° tracks at channel segments (figure 8) guarantee the spacing is enough.

The routing grid modifications for missing bumps enable octilinear RDL routing in arbitrary shaped bump arrays as long as array rows and columns are aligned. Therefore it can be applied to complex packaging designs where simple graph modeling cannot reflect the exact routing capacities in the channels and areas.

## 4.3 Overall Algorithm

By the analysis in this section, the basic RDL router can be realized by the following algorithm. The input is a bump array with $n$ lines, $m$ columns and a group of wire-bonding pads, and the output is the RDL routing result.

The flow solution can be computed by the standard min-cost max-flow algorithm, where the cost on each edge is its length based on the distance between grid nodes. In figure 7 and 9, we assign each horizontal/vertical edge with weight 10 and a full grid diagonal edge with 14. Since we use “line shift” in the graph, the actual locations of routing tracks are usually shifted from the grid nodes they connect to, and the “edge length” values are not accurate. But as long as we

can avoid spacing violations (by figure 6 and theorem 2), the network model is valid.

**Algorithm 1: RDL router**

1. Routing network graph $G \leftarrow$ ({bumps in the array}, φ);
2. Construct top horizontal channel and left vertical channel
3. For $(i, j) \leftarrow (1, 1) \sim (n, m)$
   Construct channel segment below bump$(i, j)$;
   Construct channel segment at right side of bump$(i, j)$;
4. For $(i, j) \leftarrow (1, 1) \sim (n, m)$
   if there is a bump at array $(i, j)$
   Connect the bump to surrounding channels;
   else Construct Manhattan network in the open area;
5. For $(i, j) \leftarrow (2, 2) \sim (n, m)$
   Add $(C^{i,j}_{\text{cross}} - 2)$ 45° tracks in channel crossing area between bump $(i - 1, j - 1)$ and $(i, j)$;
6. For $(i, j) \leftarrow (2, 1) \sim (n, m)$
   Add $(C^{i,j}_{\text{hseg}} - 1)$ 45° tracks in channel between bump $(i - 1, j)$ & $(i, j)$;
7. For $(i, j) \leftarrow (1, 2) \sim (n, m)$
   Add $(C^{i,j}_{\text{vseg}} - 1)$ 45° tracks in channel between bump $(i, j - 1)$ & $(i, j)$;
8. For $(i, j) \leftarrow (1, 1) \sim (n, m)$
   if there is NO bump at bump$(i, j)$
   Extend 45° tracks into the open area of $(i, j)$ from surrounding channel segments;
   Re-compute 45° tracks in channel crossing areas beside the open area of $(i, j)$;
9. Add wire-bonding pads in $G$ and run min-cost max-flow;

Another problem which may affect the solution qualities is that if the wires are optimized only on length, the algorithm may choose paths with zigzags. Especially when the bump array has some large open areas in it, the problem may cause unnecessary manufacturing difficulties. As shown in figure 10(a), there are a large number of shortest routes from point A to B in the shaded quadrangular area. The dotted one here is a shortest path, but it contains many jogs which are not desired in practical RDL routing designs. Additional post-process is needed to clean up these jogs.

## 4.4 Wire Smoothing

While the min-cost max-flow algorithm gives a solution provided with a function of edge cost, for the smoothing process we need to minimize the number of turning points as well as the wire length to avoid unnecessary zigzags. Since the number of turning points cannot be represented as a function on edges, the smoothing process cannot be embedded in the min-cost max-flow. We add it as a post-processing step on the network flow solution. The basic iterative flow is described as follows:

**Algorithm 2: Smoothing process**

Repeat {
  For each unit of flow:
    1) Delete the flow unit from source to sink;
    2) Find a shortest path with minimal number of turning points, avoiding the nodes used by other flow units;
    3) Resume the flow unit by the new path;
} Until *no path is changed* in the iteration

The key iteration is step 2), which can be implemented by a modified shortest path algorithm using dynamic programming. The minimization objective is not only the path length, but also the number of turning points in the path (with minimum path length provided). In [5], an α-β solver in O(*n*log*n*) time based on Dijkstra algorithm [4] is introduced, which can minimize path turning points on a Manhattan grid. Though not explicitly mentioned, dynamic programming technique is adopted in α-β routing to achieve the time efficiency.

We use a similar idea for octilinear routing here. For simplicity of description we use Bellman-Ford algorithm [4] as the base. We replace the distance variable by the (distance, #turnings) pair. By this replacement we find the "relax" step no longer valid, because the (distance, #turnings) pair does not contain the information where the values come from. So when relaxing edge $(u,v)$ we cannot tell whether the number of turnings should be increased.

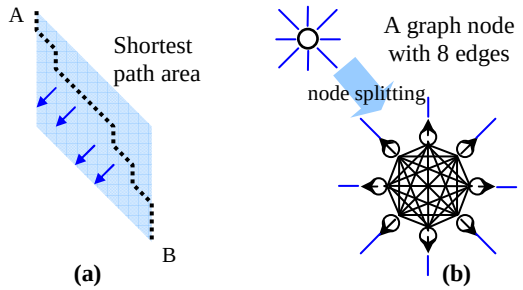
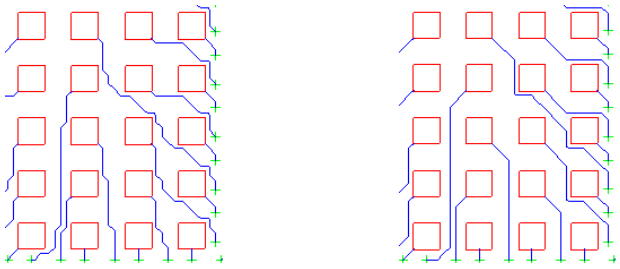



**Figure 10. Smoothing wires**

Using dynamic programming technique, we split each node in the network graph into 8 states (as in figure 10(b)), each containing a (distance, turnings) pair indicating the best path coming from each of the 8 directions. Thus, the nodes in the graph are multiplied by 8 and the information of the turning points can be obtained: If the direction of the outgoing relaxing edge is different from the incoming direction, we increase the number of turning points by 1, otherwise preserve the original value.

The actual implementation can use Dijkstra algorithm with time complexity O(*n*log*n*) here. With an efficient solver for the shortest path with minimal turnings, the whole smoothing is performed iteratively. In the iterations, we change some paths, and thus change the shape of open space beside these paths. In this way, changes on other paths are propagated in following iterations until reaching stable. Algorithm 2 does not guarantee minimum number of turnings in the final solution because of some imperfections of the grid network constructed between different types of areas. But in practice the resulting wires usually have acceptable shapes. Figure 11 shows part of an RDL routing solution, where the original wires contain a lot of zigzags and the smoothing process turned them into reasonable lines easy for manufacturing.



**(a) Before smoothing** **(b) After smoothing**

**Figure 11. Example of smoothing effect**

## 5. EXPERIMENTAL RESULTS

We implement the octilinear RDL router in C++ programming language on a 2.8GHz Intel Xeon workstation with 8GB memory and Linux operating system.

Previous RDL routing is mostly done manually, and therefore standard benchmarks are unavailable. We construct test cases artificially as well as from real industrial designs.

In our tests, we first snap all the wire-bonding pads to the nearest grid nodes. Since the pads are usually small – only slightly larger than a grid – the routing solution for snapped pads can be used directly. If the pads are placed far away from the bump array, clearly no spacing violation will happen between wire and pad; Otherwise, like the case in figure 13 & 14, we need to take extra care of the network components around the pads so that the wire tracks do not conflict with the pads. In most cases the pads are placed in lines, so deleting some Manhattan and 45° track segments will be sufficient. In a few corner cases, we use some extra heuristics to modify local network components in order to preserve enough spacing.

Figure 12 shows a small bump array with open space in the middle and two different distributions of pad arrays. Two large cases derived from real designs are shown in figure 13 and 14. The effectiveness of the algorithms and the overall flow is shown in the RDL routing details.

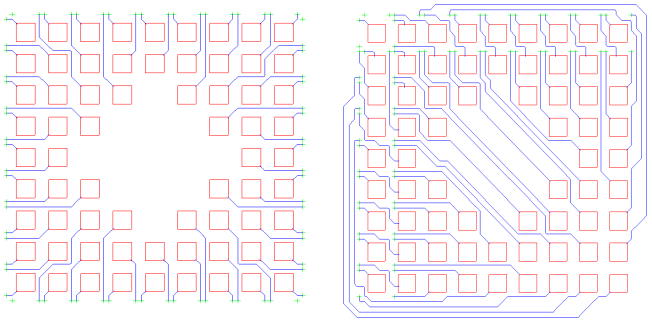


**Figure 12. Results of two pad distributions in one bump array**

## 6. CONCLUSIONS AND FUTURE WORKS

We propose the first octilinear RDL router based on a grid network and the "line shift" technique. We show that for ideal full bump array and wire-bonding pads snapped to grid, our router can always give the best solution. The wires are optimized by total length with small shift errors and are smoothed into reasonable shapes for practical applications.

The model and algorithm we use are proved to be effective for RDL routing, while still with some limitations. For instance, only single layer routing can be performed in our algorithm. In some cases we may need multiple layers because of high routing congestion, and some methodologies in [6] can be adopted for multi-layer RDL routing in future works. Also the fractional capacity loss in section 4.2 may affect routability in our solution. Alternative models to avoid capacity loss are to be explored.

## 7. REFERENCES


[1] M.-F. Yu, J. Darnauer, W.-M. Dai, "Interchangeable pin routing with application to package layout", IEEE/ACM Int. Conf. on Computer Aided Design, pp.45-50, 1996.

[2] J.-W. Fang, I.-J. Lin, P.-H. Yuh, Y.-W. Chang, J.-H. Wang, "A routing algorithm for flip-chip design", IEEE/ACM Int. Conf. on Computer Aided Design, pp.99-104, 2005.

[3] J. Hershberger, S. Suri, "Efficient breakout routing in printed circuit boards", Annual Symp. on Computational Geometry, pp.460-462, 1997.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, "Introduction to Algorithms, Second Edition". MIT Press and McGraw-Hill, 2001.

[5] T.-C. Hu, M.-T. Shing, "The α-β routing", VLSI Circuit Layout: Theory and Design, pp.139-143, IEEE Press, 1985.

[6] R. Wang, R. Shi, C.-K. Cheng, "Layer minimization of escape routing in area array packaging", IEEE/ACM Int. Conf. on Computer Aided Design, 2006.

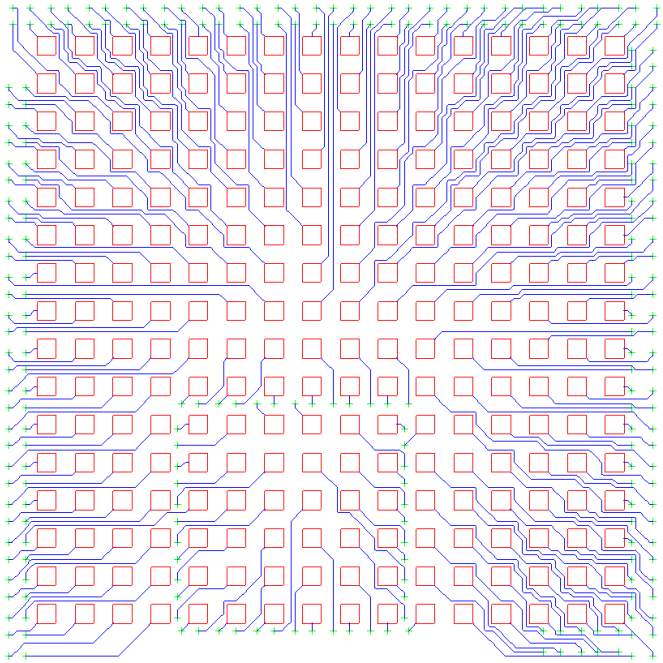



**Figure 13. Experimental results of the octilinear RDL router**

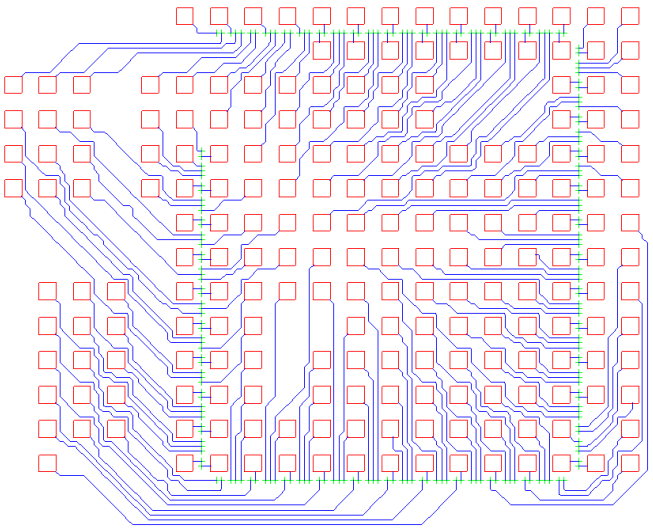


**Figure 14. Experimental results of the octilinear RDL router**