Irregular Bumps Design Planning for Modern Ball Grid Array Packages

Hsin-Yu Chang, Jyun-Ru Jiang, Simon Chen, Hung-Ming Chen and Ya-Ying Chien
Institute of Electronics, National Chiao Tung University, Taiwan
e-mail : sandychang123@gmail.com

Abstract— In modern flip-chip packages, bumps are often placed irregularly due to different design needs. It costs a great amount of time and manual effort to generate substrate routing from bumps through vias to package balls. Moreover, any single model in prior works could not be simultaneously applied between bumps, vias and balls. In this work, we propose a unified flow network model to formulate the 2-layer substrate routing problem on irregular package structure. We present a renovated bump model that can handle irregular bump plans, filling the gap/insufficiency in existing models. With our methodology, signal assignment on vias and balls, and substrate routing on two layers can be obtained at the same time. We also present an iterative optimization technique to further improve wire congestion. Our results show that the proposed method completes via and ball assignment efficiently, and obtain 100% routability while improving 16.45% in wirelength, compared with manual design in real industrial cases.

I. Introduction

In modern flip-chip package design, there are hundreds of signals between chip, package, and board domains. Due to the increase of complexity and I/O pin amount, bumps on redistribution layer (RDL) and balls in ball grid array (BGA) package have become a major interface between the chip and the printed circuit board (PCB). In order to accelerate the design speed, a package designer needs to deliver the signals on the balls and the corresponding substrate routing result as soon as the chip information is determined. Signal assignment and routing plays an important role. As illustrated in Fig. 1, a signal from chip travels from bumps through vias and balls on the package substrate to the PCB. In current industrial design flow, the assignment was first generated by designers manually based on their experiences, then performing substrate routing according to the initial assignment. However, such initial assignment lacks of routing information, especially due to the irregularity of bump positions. Serious net congested problems might happen and re-assignment and rerouting become ordinary. This back-and-forth process usually takes two to three weeks and may cause a great amount of manual efforts.

A. Previous Works

Several previous works proposed methods to solve signal assignment and global routing problem. [1] proposed an approach that incrementally improves the via assignment and generates a global routing result. In [1], it is assumed that signals start from bonding fingers, which are placed as a single ring of signal sequence aligned neatly

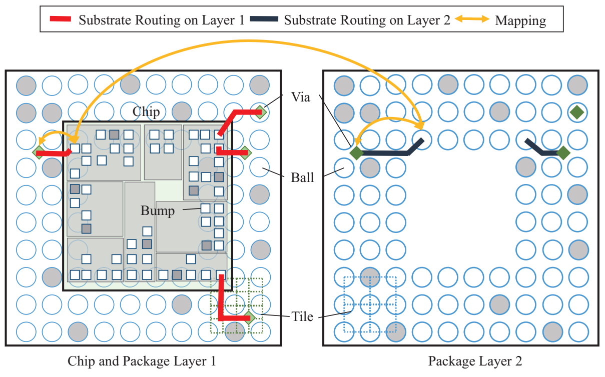


Fig. 1. Top view of chip and package. The left figure shows a chip on the package and the substrate routing on the top layer of the package. Bumps are placed irregularly around the chip. The blue circles are balls and the green diamonds are vias, which are regularly aligned on the package. The right picture shows the bottom layer of the package. If a via is placed on the ball, substrate routing is done automatically for the bottom layer.

around the chip boundary, instead of an irregular bump array. Also, in [1], signal assignment on vias needs iterative adjustment to obtain higher routability. [2] addressed an auto-assign method on package balls based on decreasing fly-line crossings. [3] also developed a similar assignment method based on fly-line reduction but focuses on bump assignment. However, for [2] and [3], judging routability only on number of crossing fly-lines may not be accurate. In addition, most often used method such as [1] and [3] partitions the model into four portions. Although partition can decrease the complexity of the problem, it will reduce solution space which can easily cause an unroutable solution. As shown in Fig. 1, bump arrays may form in multiple rows and be placed off-grid, so substrate routing should also consider routes between bumps. However, there are no model in previous works that considers irregular bump positions. To achieve a via and ball assignment with high routability, a new model should be created to solve the problem.

B. Our Contributions

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first work that can handle routing problem on irregular package structure by using a unified flow network model.
- A new irregular bump plan model that can formulate the routing between off-grid bumps and on-grid balls efficiently.
- Our proposed model can simultaneously assign sig-

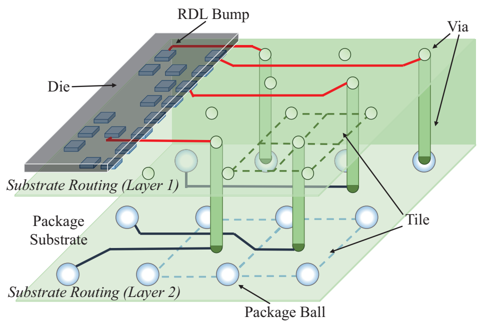


Fig. 2. A detailed cross section of our flip-chip. In this work, signal passes from bumps to vias through substrate routing on layer 1 (red routes), and then from vias to balls through substrate routing on layer 2 (black routes).

nals of bumps to vias/balls and generate an optimized routing result.
- An optimization scheme to minimize the maximum congested region and further obtain routing results with better wire congestion distribution.
- Experimental results on four real industrial cases show that our methodology achieves 100% routability without design rule check (DRC) violations, and results show an average wirelength improvement of 16.45%, compared with manual designs.

The rest of this paper is organized as follows. Section II defines the problem of our signal assignment and routing problem. Section III introduces our algorithm flow and describes our proposed flow model. Section IV describes our detailed routing algorithm; Section V shows the optimization method of flow congestion. Section VI reports our experimental results on four industrial cases, followed by conclusions in Section VII.

II. Problem Formulation

Fig. 2 shows a sketch of our two-layer package model. We are given the physical location and the signal of each bump, which is fixed in our cases. Each signal starting from a single bump will be connected to a via on the top of the package substrate layer (Layer 1), and each via will then be connected to a ball on the bottom of the package substrate layer (Layer 2). A via on Layer 1 can be placed directly at the spot corresponding to a ball's position on Layer 2, or between four adjacent balls. We define a tile on Layer 2 a rectangle between four adjacent balls, and a tile on Layer 1 one-out-of-four of a tile on Layer 2.

Our ball and via assignment problem is to simultaneously find a solution of via assignment on the first layer, the ball assignment on the second layer and the corresponding routing results on two layers.

The formal definition is as follows:

Input
- Physical locations of bumps and balls on package
- Signals on each bump pad
- Design rules such as routing spacing constraints, size of bump, via and ball

Output
- Assignment of signals on balls
- Assignment of signals and physical locations of via
- A corresponding planar substrate routing result on two layers

Objectives
- 100% routability with minimized wirelength

III. Simultaneous Global Routing and Ball/Via Assignment

We introduce our design flow in Sec. A, and a via moving method in Sec B. Then, we will describe the three different routing models we use in our hybrid model, including the new bump model on Layer 1 (Sec. C), the stagger via model on Layer 1 (Sec. D), and the ball model on Layer 2 (Sec. E). Finally, we will explain the connection between each model in Sec. F.

A. Design Flow

Our algorithm consists of four stages: (1) forbidden area via moving, (2) model construction, (3) global routing and via assignment, and (4) detailed routing. In the first stage, we perform a fast method to move via away from forbidden areas. Then, based on the moved via position, our hybrid flow model is constructed in the second stage. The third stage applies the obtains the ball/via assignment and the substrate routing result simultaneously. Finally, detailed routing on two layers will be applied.

B. Forbidden Area Via Moving

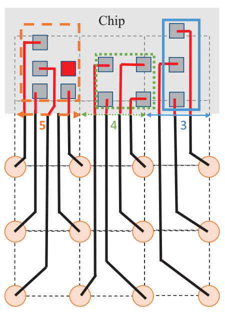
As we can see in Fig. 2, if a via is placed on a ball of Layer 2, the signal can be assigned directly to the ball and does not require routing resources on Layer 2. Otherwise if a via is placed between four adjacent balls, routing on Layer 2 is needed. Since the routing resources on Layer 2 are more scarce, to obtain least wire length on Layer 2, we will first assume that all of the vias are placed on a ball. However, there may exist *forbidden areas* where vias can not be placed, such as areas overlapping with the chip, or regions covered with power and ground meshes and special components. To move the vias away from those *forbidden areas* without increasing much wirelength, we will first perform a fast *forbidden area via moving* method as follows.

We model the via moving problem as a weighted bipartite matching problem. The vias in forbidden areas that must be moved away is referred as *to-be-moved vias*, and legal positions that the vias can be moved to is referred as *candidate positions*. We construct a flow network where each *to-be-moved vias* and *candidate positions* corresponds to a vertex, and the edge cost between two vertices is the Euclidean distance between the two positions. Also, to decrease the number of edges, if the distance between a *to-be-moved via* and a *candidate position* is too far, an edge will not be constructed. By applying the MCMF algorithm [4] we can find a bipartite matching solution on the flow network, and obtain new via positions for each *to-be-moved via* without increasing much wirelength on Layer 2.

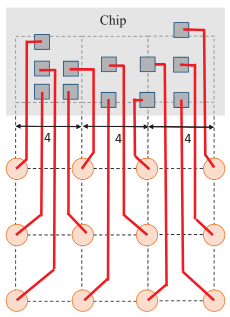
C. Layer 1 Bump Model

Bumps are placed according to the placement of macros inside the chip, since the macros varies in sizes and positions, the position of bumps are not very regular. As shown in Fig. 1, although balls are aligned regularly and can be cut into unit tiles easily, bumps not only are placed irregularly but also do not align with the balls. However, there are no prior models that considers irregular bump positions, and using a single model on balls and bumps is impracticable.

We explain our motivation of this study in the following example. A similar routing problem that connects inter-tile vertices to I/O pads outside the boundary is



(a) Estimate routing without considering bump position



(b) Estimate routing considering bump position

Fig. 3. An example when estimating routing without considering actual bump positions. In (a), edge capacity is first estimated, then escape routing is preformed from ball to bump. The red bump could not be routed. While considering the actual position of bumps, a routing result can be found in (b).

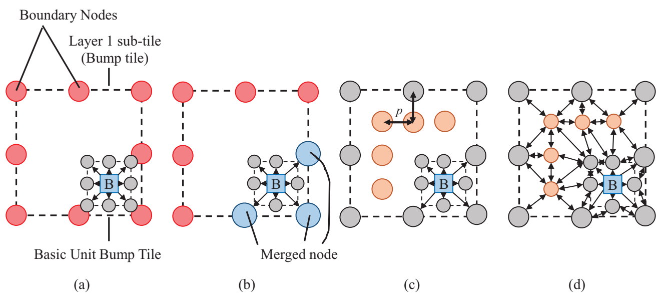


Fig. 4. Process of creating the bump routing model. (a) A basic bump tile and boundary nodes. (b) Node merging. (c) Extra node adding. (d) Edge connection.

mentioned in [5]. [5] uses an estimating method of adjusting the capacities of the boundaries to control the number of nets passing through. However, if the density of bumps is unbalanced and larger near a boundary than others, some of the bumps may need to escape from another boundary, rather than the estimated one. The red bump in Fig. 3(a) could not escape the estimated boundary so routing has failed. However, in Fig. 3(b), the spacing is actually enough if the bump is allowed to escape through the neighboring boundary. Routing spacing between bumps should be checked globally to get a correct routing estimation. In order to correctly model the net capacity and routability between bumps and to create a complete flow model from bump to ball, a new bump model is proposed as follows.

Pitch p is the minimum distance that a net can be routed next to a bump or next to another net. First, a basic *unit bump model* will be created wherever there is a bump, as shown in Fig. 4(a), the model at bottom-right is a *unit bump model*. The center node of the *unit bump model* is the bump, and the rest eight nodes represent positions where the signal can be routed to, the distance between each nodes in the *unit bump model* is equal to or greater than p. Then, since bump positions are not regular, *bump-tile* is created to align to a tile on Layer 1. The larger rectangle in Fig. 4(a) is a *bump-tile*, we will add eight *boundary nodes* on the edges of the *bump-tile* (the red nodes in 4(a)). If the distance between two nodes is smaller than p, the two nodes will be merged into one, to

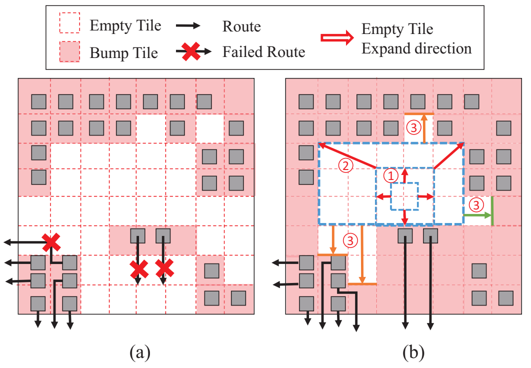


Fig. 5. Process to reduce the *bump-tiles* in the center area. (a) Only the tiles with bump were set as bump tile. (b) Expand empty tile from center: (1) Rectangular expanding, (2) Corner expanding, (3) Side expanding.

ensure that distance between two nets will not violate the design rule. The three blue nodes in Fig. 4(b) were the merged nodes. Moreover, to model the routing resources between bumps, extra nodes will be added at positions where the spacing between other nodes is greater than the p. The orange nodes in Fig. 4(c) are the added extra nodes. Finally, in order to obtain a planar routing result, the edges between the nodes will be created by the Delaunay triangulation method[6] as shown in Fig. 4(d).

Bumps will only be placed in the chip area, so all tiles on Layer 1 inside the chip could be constructed as a *bump-tile*. However, it is not realistic since we need to have some bumps for power/ground. Routing near the center should be avoided, so we need a process to reduce the *bump-tiles* in the central area. On the other hand, if we only set the tiles overlapped by bumps as *bump-tiles*, some bumps may not be able to find routing solutions, such as the example shown in Fig. 5(a). The bump in the bottom-left corner and the two bumps locating near the center could not find routing paths due to the lack of bump tiles. One way to generate practical bump-tiles is to set tiles from outer to inner space, but even if some outer space does not overlap with bumps, we still cannot determine whether there is bump near the center. So instead of setting tiles from outside, we expand empty regions from the inside. We propose an *empty region expanding methodology* that will decide *empty tiles* without surrendering the routing resource.

An example of our process is shown in Fig. 5 (b), we will initially set all tiles inside the chip to *bump tiles*, and then expand the empty area from the center of chip outward. There are three steps to expand the empty area: (1) Rectangular expanding, (2) Corner expanding, and (3) Side expanding. First, an initial empty rectangular area will be set in the center of the chip, and *Rectangular expanding* will be performed iteratively by expanding the four sides of the rectangular area with same unit step (for example, the width of a tile), until the rectangle reaches a bump. Then, *Corner expanding* will further expand the four corners of the rectangle separately. By *Rectangular expanding* and *Corner expanding*, a maximum rectangular empty area inside the chip could be found. Finally, in order to maximize the empty area, the *Side expanding* process is to expand each of the four sides in order to find the contour of bump. Also, if expanding reaches the boundary of the chip, we will reserve a distance from boundary as the bump tile region to ensure the routing resource for bumps close to the tile boundary.

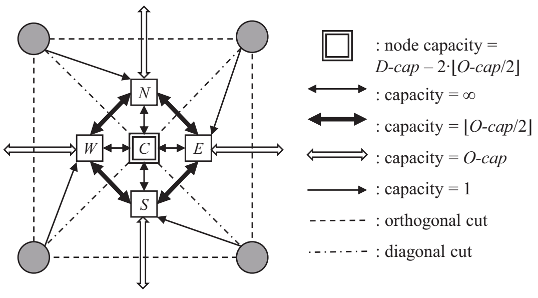


Fig. 6. The network flow model inside a tile proposed by [7]. A *tile* is formed by four adjacent pins in this model, while in our structure, there may exist an extra pin in the center of the tile if a via is placed.

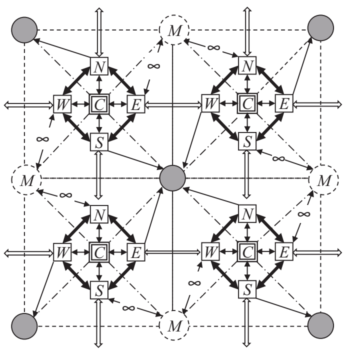


Fig. 7. Our modified model for Layer 1 inside a tile, the original tile is split into four sub-tiles. If a via is placed in the middle of the tile, the middle node will be set as a regular pin node. Otherwise, the middle node will be set to a missing node. Also, the direction of edges connecting to the pins are opposite to [7].

D. *Layer 1 Stagger Via Model*

A network model that can correctly model the orthogonal capacity constraints and the diagonal capacity constraints between two adjacent pins is proposed in [7]. As shown in Fig. 6, a *tile* is formed by four adjacent pins. The maximum number of nets that can pass through the sides of each tile is denoted as *O-cap*, and the maximum number of nets that can pass through the diagonals of the tile is denoted as *D-cap*. For the vias of Layer 1, we construct our model based on [7] due to the similar properties of edge capacity, and modifications were made to fit our structure.

In our flip-chip structure, if all vias are placed on a ball, the vias will form a regular pin array with capacity constraints limiting the number of wires that can pass through two adjacent vias, which is very similar to the proposed model in [7]. However, since a via can also be placed in the middle of a tile if needed, we renovate the model of [7] to a stagger via structure shown in Fig. 7. The original tile will be partitioned into four sub-tiles. If a via is placed in the middle of the tile, the middle node will also be set as a pin node. Otherwise, the middle node will be seen as a missing node, and will be replaced by a *missing pin* with node capacity *delta-cap*, which is also proposed in [7]. Also, the direction of edges connecting to the pins are opposite to [7] because that the source of our flow network starts from the bumps, instead of the vias.

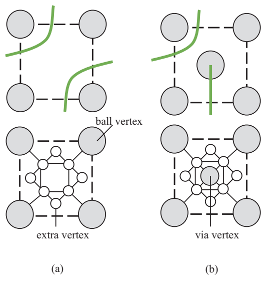


Fig. 8. The network flow model proposed by [8]. (a) routing result when there is no via placed in the grid, (b) situation when a via is placed in the grid.

E. *Layer 2 Ball Model*

The size of balls on Layer 2 is much bigger than vias on Layer 1, and the number of nets that can pass through a tile is also much less. We adopted a simpler model proposed in [8] for Layer 2. As shown in Fig. 8, the routing graph has three kinds of vertices. The ball vertex and the via vertex represents to ball and via respectively, and the extra vertex models the routing in this graph. Because that the ball radius is relatively larger than the width of a net, the capacity of every extra vertex is one. Fig. 8(a) shows a routing result of the situation when there is no via placed in the grid, and Fig. 8(b) corresponds to the situation when a via is placed in the grid. Also, the via vertices in the Layer 2 model are connected to the corresponding via vertices of the ball model on Layer 1. We will further explain our complete flow graph connection in the next subsection.

F. *Flow Model Connection and Ball Assignment*

After constructing the three different models, we combine the models together to form our complete flow model. An overview can be seen in Fig. 9. The bump model and the stagger via model will be connected through the corresponding boundary nodes, which is the red nodes in Fig. 9. If a tile of the stagger via model is adjacent to a bump tile by the west boundary, the west node of the stagger via model will be connected to the middle node at the east boundary of the bump tile. The upper-left node of the stagger via model will be connected to the upper-right node of the bump tile, and so on. In order to preserve the edge capacity on the boundary edges, if the node on ball side is N, E, W, or S, the capacity of the edge between the vertices will be *O-cap*. If the node on the ball side is a missing node, the edge capacity of the connection will be *delta-cap*, which is the capacity of missing node defined in [7]. Otherwise, if the node is a via, the edge capacity is one.

The super-source will be connected to each bump nodes in bump model on Layer 1, and the edges are assigned capacity 1. The super-sink is connected to each ball nodes in the ball model on Layer 2, also with edges of capacity 1. The stagger via model on Layer 1 is connected to the ball model on Layer 2 by the corresponding via nodes and ball nodes. After the connection, we apply minimum-cost maximum-flow (MCMF) algorithm [4] to obtain the routing result. If the capacity of the flow network is enough, the MCMF algorithm will provide a routing result with the minimal wirelength. Then, by tracing the flows starting from each bump, we can simultaneously obtain the routing path of each signal, and also obtain the assignment via and ball.

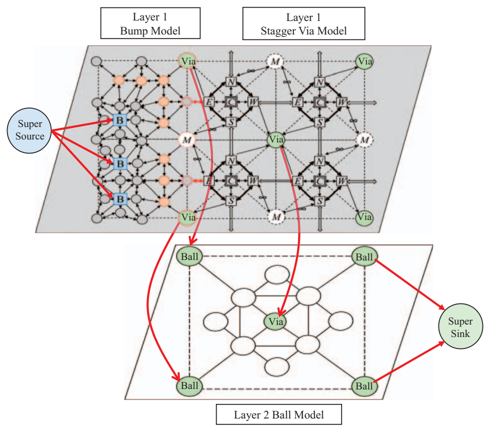


Fig. 9. An overview of our whole flow model. The super source is connected to all the bump nodes in Layer 1 Bump Model. The bump model is connected to the stagger via model with the red boundary nodes. The via nodes in Layer 1 model are connected to the ball and via nodes in Layer 2 model. The ball and via nodes in Layer 2 model are connected to the super drain.

IV. Detailed Routing

In our detailed routing algorithm, we apply flow decomposition proposed in [7]. After flow decomposition, we have a planar topological routing, representing by a set of path $X = \{\pi_i \mid i \in N\}$, where N is the set of all nets. Our objective is to find another topologically equivalent set of path $\bar{X}$, where the minimum spacing between all paths and the minimum spacing between paths and obstacles (bumps and vias) in $\bar{X}$ meets our requirement. First, we apply Delaunay triangulation [6] to all the obstacles. Second, we find all the intersection points of all the paths and the triangulation edges. Then we can represent paths by the intersection points. For each triangulation edge, if there exists an adjacent pair of intersection points of a same path, meaning the path cross through the same triangulation edge twice. We can reduce detour of the path by iteratively removing such pair of intersections. To generate enough spacing between paths, also between paths and obstacles, we iteratively push intersection points on each triangle when there exists a spacing violation. In addition, a post-processing is performed to reduce sharp turning.

V. Max Flow Congestion Minimization for Routability

A design with regions of high net congestion may worsen the performance of such area. From the method proposed in Section III, a via and ball assignment solution and a routing result with the shortest wirelength could be obtained. However, net congestion could not be considered simultaneously. In order to further minimize the net congestion of the routing result, we propose a *max flow congestion minimization method* to minimize the max flow of the global routing result. If a flow enters a node of a tile, and leaves the tile from another node of the tile or arrived at the via node inside the tile, it *passes* the tile. The *flow amount* in a tile is the number of flow passing the tile. Because of the routing constraints and the structure of the flow model, the maximum flow amount of tiles on Layer 2 will not be greater than 2. Therefore, we will only perform this optimization on Layer 1.

The proposed optimization scheme is to iteratively (1)

Algorithm 1: Flow congestion minimization algorithm

```
Input: #flow in each tile $f_i$, upper bond $thres$
Result: global routing result
1  while exist $f_i$ >$thres$ or $iteration_t$ <$maximum_it$ do
2      for each tile $i$ do
3          if $f_i$ >$thres$ then
4              $cost_i$ += ($f_i$ - $thres$) ;
5          end
6          if $f_i$ <$thres$ then
7              $cost_i$ -= 1 ;
8          end
9      end
10     Run global route with new cost again ;
11     if max $f_i$ <$thres$ then
12         end while loop ;
13     end
14 end
```

Add penalty to congested tiles and (2) Lower down the added penalty if the congestion is improved. Before performing the algorithm, an expected threshold flow amount ($thres$) is defined. The objective of our algorithm is to modify the cost ($cost_i$) of each tile i such that the flow amount (f_i) of all tiles are less than $thres$. If f_i in a tile is greater than $thres$, we will increase the cost of all edges in the tile by (f_i - $thres$), in order to restrict the flows to pass through such tile. Also, if f_i of a tile decreased through iterations, cost on such tile should be decreased. However, since decreasing the cost of such tile might cause the tile over-congested again, we will only decrease the cost by one. We define a tile t_i, and the flow amount in t_i $flow_i$. The detailed processes are shown in Algorithm 1.

VI. Experimental Results

We applied the proposed algorithms on four real and large scale industrial cases. All of the cases contains a single chip with bumps placed in irregular positions. Since no previous model could handle irregular bump position, we implemented a method combining initial via assignment and maze routing called INIT to compare with our model. INIT is to first assign signals from bump to each via decided using the weighted bipartite matching algorithm. Then, INIT orders the signals by the position relative to the center point of the chip clock-wisely, and apply maze route to each signal in order. However, INIT can only achieve routability lower than 90%, while ours can achieve 100% routability in all cases with the same design constraint. Table I listed the routability and wirelength of the manual routing result routed by designers and our method. Our runtime is much less comparing to the manual effort, and our wirelength can also achieve a better quality comparing to manual's. The detailed routing results are shown in Fig 10. By using our detailed routing algorithm, we can achieve a routing result without DRC violations.

Table II shows that our congestion optimization method can decrease the maximum flow number in short iterations on all four industrial cases. The WL listed in Table II is the global routing wirelength on Layer 1 because we only apply the method on Layer 1 during global routing step. The wirelength of all cases is slighted in-

TABLE I
Global Routing comparison between our model and manual routing result. ("#sig.": number of signals in the case. "WL": total wirelength. "Rout.":Routability. "Time": algorithm runtime or manual man-hours. "improv.": Wirelength improvement between manual and our method.)

Testcase	#sig.	Manual			Ours			
		Rout.	Time	WL(μm)	Rout.	Time(sec)	WL(μm)	improv.
Case1	387	100%	2-3 weeks	843168.5	100%	8.3	669301.5	20.6%
Case2	340	100%	2-3 weeks	948696.5	100%	6.5	764961.8	19.4%
Case3	379	100%	2-3 weeks	793431.8	100%	8.5	681850.7	14.1%
Case4	380	100%	2-3 weeks	1206129.3	100%	25.8	1065578.1	11.7%

TABLE II
Flow Congestion Optimization Results. ("#sig.": number of signals in the case. "max flow": max flow amount and number of tiles with max flow. "avg": average flow amount. "WL": global routing wirelength of Layer 1.)

Testcase	#sig.	#tiles	Initial			After			#iter
			max flow	avg	WL(μm)	max flow	avg	WL(μm)	
Case1	387	728	5 (3 tiles)	0.789	666502.7	4 (104 tiles)	0.781	669037.5	1
Case2	340	728	5 (5 tiles)	1.001	822278.4	4 (146 tiles)	1.008	825192.6	13
Case3	379	728	5 (2 tiles)	0.826	696129.8	4 (121 tiles)	0.831	697785.7	1
Case4	380	1024	5 (3 tiles)	0.898	1089005	4 (150 tiles)	0.904	1095052	7

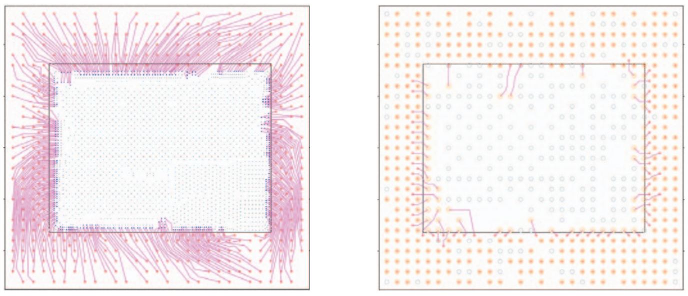

(a) Layer 1 (b) Layer 2

Fig. 10. The 2-layer substrate routing solution of Case1. The blue squares in (a) are the bumps, the red circles are the vias, and the orange circles are the balls. The purple segments are the routing result.

creased because our method modified the edge cost of the model so the MCMF solution will not be optimal for wirelength.

VII. Conclusions

In this paper, we introduced a renovated and unified flow model to formulate the 2-layer substrate global routing, to solve the problem that single model in previous works could not handle in irregular package structure and routing between bumps, vias and balls. We proposed a new bump model to handle the routing between irregular bumps. Our model can simultaneously obtain the signal assignment on balls and vias, and generate an optimized routing result on two layers. We also presented a detailed routing algorithm that can achieve a routing result without DRC violations. Package designer can obtain an assignment result with 100% routability efficiently. In addition, we provided a heuristic for better wire congestion distribution. The experimental results have shown that our method can achieve 100% routability on four industrial cases, and is much faster and has better wirelength performance than the traditional manual flow. We plan to handle electrical constraints such as EM and SI in future work.

References

[1] Y. Kubo and A. Takahashi, "Global routing by iterative improvements for two-layer ball grid array packages," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 4, pp. 725–733, April 2006.

[2] H. Han, W. Yin, W. Wang, and Z. Pang, "Auto-assign method for large scale flip-chip package design," in *9th IEEE International Conference on ASIC*, Oct 2011, pp. 929–932.

[3] R.-J. Lee and H.-M. Chen, "A study of row-based area-array i/o design planning in concurrent chip-package design flow," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 2, pp. 30:1–30:19, Apr. 2013.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[5] B. Q. Lin, T.-C. Lin, and Y. W. Chang, "Redistribution layer routing for integrated fan-out wafer-level chip-scale packages," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2016, pp. 1–8.

[6] S. Fortune, "Handbook of discrete and computational geometry," J. E. Goodman and J. O'Rourke, Eds. Boca Raton, FL, USA: CRC Press, Inc., 1997, pp. 377–388.

[7] T. Yan and M. D. F. Wong, "Correctly modeling the diagonal capacity in escape routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 2, pp. 285–293, Feb 2012.

[8] Y. Tomioka and A. Takahashi, "Routability driven modification method of monotonic via assignment for 2-layer ball grid array packages," in *Asia and South Pacific Design Automation Conference*, March 2008, pp. 238–243.