On Pre-Assignment Route Prototyping for Irregular Bumps on BGA Packages

Jyun-Ru Jiang*, Yun-Chih Kuo†, Simon Yi-Hung Chen†, and Hung-Ming Chen*
*Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan and †MediaTek Inc., Taiwan
emails: p57l63p9c69@gmail.com, Eric-YC.Kuo@mediatek.com, simonYH.Chen@mediatek.com, hmchen@mail.nctu.edu.tw

***Abstract*—In modern package design, the bumps often place irregularly due to the macros varied in sizes and positions. This will make pre-assignment routing more difficult, even with massive design efforts. This work presents a 2-stage routing method which can be applied to an arbitrary bump placement on 2-layer BGA packages. Our approach combines escape routing with via assignment: the escape routing is used to handle the irregular bumps and the via assignment is applied for improving the wire congestion and total wirelength of global routing. Experimental results based on industrial cases show that our methodology can solve the routing efficiently, and we have achieved 82% improvement on wire congestion with 5% wirelength increase compared with conventional regular treatments.**

I. Introduction

In current flip-chip packages, there are hundreds of requried I/O pins between bumps, balls and package board. The signal bumps on modern flip-chip packages often place irregularly for different design consideration. Fig. 1 shows the interface between chip and PCB board, the signal will travel from the bumps through vias to balls on the package. In the industrial flow, the signals from bump to ball are assigned manually. However, the assignment lacks of routing resource consideration, some of nets could not be routed to the assigned ball due to serious congested problem. Therefore, it needs to re-assign the balls and re-route the nets several times until all net connection is completed without any violation. It may take much time and effort to complete the package design. Hence, this paper propose an an efficient router to evaluate the quality of an assignment.

A. Previous Works

Several previous works are proposed to solve the routing problem on package and board. We can classify these works into different points of view: *assignment* type and *pin placement*. From the *assignment type* perspective, the assignment type of flip-chip routing problem can roughly divide into two categories: **pre-assignment** and **free-assignment** problems. In pre-assignment routing problem, the signal from signal bumps to balls are predefined by IC designer or package designer before routing. In contrast, free-assignment routing problem can assign each bump to each ball freely when routing the nets.

Free-assignment problems are often solved by network-flow-based algorithms, such as [9]. For the pre-assignment problem, it can be solved by integer linear programming(ILP) like the work in [1], or other heuristics method like the works in [3], [4], [6], [8]. In [3], [4], they applied an iterative via assignment improvement to minimize the wire congestion and total wirelength. [6] used a method based on routing sequence exchange to derive a global routing efficiently. In [8], they applied an A* based routing method with dynamic pushing to handle the net order problem. Other kinds of problem combined the free-assignment and pre-assignment problem, in previous works they often applied two-stage approach, for example [2], [5], [7].

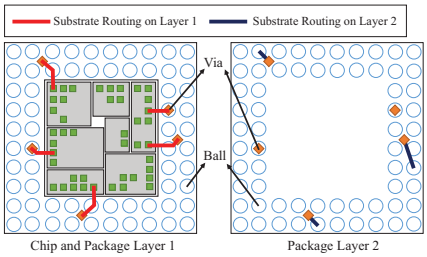


Fig. 1. Top view of chip and package on irregular bump plan. The green squares are the bumps which placed irregularly around the chip.

From the *pin placement* perspective, most of the previous works considered on-grid pin placement, including [5]. Another work [9] can handle staggered pins. However, to the best of our knowledge, none of the previous works can handle routing problem on irregular package structure. Most of the previous works exploit a specific structure of pin placement, while our approach can handle arbitrary pin placement.

B. Motivations

In practical cases, the bump placements are irregular due to the macros inside chip, varied in sizes and positions (Fig. 1). Additionally, the pre-assignment connection near the same boundary may be assigned to different side due to some design considerations. Through a unified grid based algorithm to handle these irregular bumps is not enough.

In an on-grid based routing algorithm, like [5], they used an estimating method of adjusting the capacities of the boundaries to control the number of nets passing through. Since the density of bump is unbalanced, some bumps need to escape their signal from another boundary, rather than assigned one. As shown in Fig. 2(a), the red bump is unable to escape its signal to nearest boundary, it needs to escape to other boundary with enough space. However the spacing between bumps is actually enough to pass through a wire as shown in Fig. 2(b). Since the on-grid pad placement based algorithm cannot estimate the capacity between irregular placed bumps on real cases, we need to apply a better method and model to deal with irregular package structure.

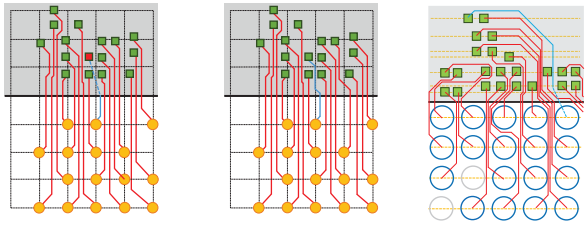
In another work [6], the congestion between the irregular bumps may have spacing violation. Fig. 2(c) shows that it has serious congestion problem due to the unbalanced bump density. Also, some nets cannot connect to the assigned ball because of the path blocked by other wire. In [6], it needs a method to adjust the assignment and the model to handle the unbalanced density bumps.

C. Our Contributions

The main contribution of this paper is summarized as follows:

- Compared to prior works, this work can effectively handle the irregular pin placement packages by using a two-stage

978-3-9819263-4-7/DATE20/©2020 EDAA



(a) (b) (c)

Fig. 2. Motivating examples when estimating routing. (a) is a estimate routing result without considering bump position. (b) shows the spacing between bumps is enough to pass a wire in practical case (red bump in (a) is escaped). (c) is an example of irregular plan trying to solve using the work in [6], the unbalanced bump density causes the routing having congestion problem.

approach. We propose a tree structure which effectively records all paths and prevent any path from passing through a same node in multi-capacity edge graph.
- Through our methodology, we can simultaneously optimize total wirelength and maximum congestion for irregular pin placement packages.
- Experimental results on real industrial cases show that our router achieved 82% improvement on wire congestion with only 5% wirelength increase, compared to previous treatments.

The rest of this paper is organized as follows. Section II defines the routing problem. Section III introduces the first part of our algorithm: escape routing for irregular bumps. Section IV describes second part: the via assignment method to optimize wire congestion and wirelength. Section V reports our experimental results on industrial cases. Finally, our conclusions are presented in Section VI.

II. Problem Formulation

In this paper, the connection between bumps and balls are predefined by IC designer or package designer. We are given the physical location and the signal of each bump, which are fixed in our case. Each signal starting from a signal bump will be connected to a via on the top layer of the package substrate layer (Layer 1), and each via will then be connected to a assigned ball on the bottom of the package substrate layer (Layer 2). A via on Layer 1 can be placed directly at spot corresponding to a ball's position on Layer 2, or between four adjacent balls. Our via assignment and routing problem is depicted as follows:

Input
- Physical locations of bumps and balls on package
- Signal on each bump pad
- Assignment of signals on balls
- Design rule such as routing spacing constraints, size of bump, via and ball

Output
- A corresponding planar substrate routing result on two layers

Objectives
- Minimize the total wirelength and wire congestion.

III. Escape routing

To resolve the routing problem on irregular package structure, we divide our methodology into two parts. The first part of our methodology is escape routing for irregular bumps, which can be split into three parts: (1) substrate routing graph (SRG) construction (2) escape point assignment and (3) A* search on SRG.

A. Substrate routing graph (SRG) construction

At the first part of escape routing, we apply a constraint triangle mesh Delaunay triangulation (CDT) with adding uniformly spreading points to build a particle-insertion-based CDT (PCDT) graph to be our substrate routing graph (SRG), in the same way as in [8]. We build a PCDT graph based on bump points, ball points and particles, then construct a dual graph of PCDT (D-PCDT) according to the PCDT.In this work, all the nets search their path on D-PCDT graph.

B. Escape point assignment

In manual pre-assignment connection, most balls are assigned outside the chip region. For these nets, we will escape the net to the on-grid tile boundary. However, some balls are assigned inside the chip region due to insufficient space or other design considerations. To deal with these inner assigned ball, we divide these balls into two types: *routable inner ball* and *unroutable inner ball*. The main distinguishing conditions of two types is whether a via could directly be put on the assigned ball without any violation. For routable inner ball, we route them inside the chip region by using A* search algorithm, and directly put their vias above their balls. For unroutable ones, we escape them to the defined boundary.

Since our escape routing is based on A* search, we need a target for cost evaluation. Hence, we need to assign an escape point for each signal bump. All the points near the on-grid tile boundary are candidate escape points, we use the following cost function to determine the candidate escape point's cost for each bump.

$$cost(b,e) = \alpha \times dist(b,e) + (1-\alpha) \times dist(B,e) \quad (1)$$

where *b* is the position of bump pad, *e* is the position of escape point candidate, *B* is the position of assigned ball, *dist(b,e)* is the distance from signal bump to escape point, *dist(B,e)* is the distance from assigned ball to escape point, and α is user defined parameter.

For those bumps need escape points, each bump will be assigned to a current smallest cost escape point one by one according to its ascending coordinate.

C. A* based escape routing on SRG

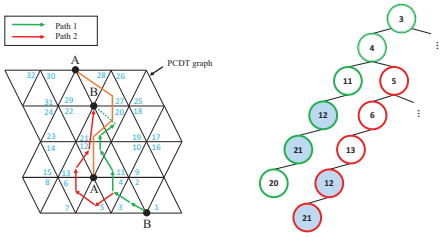
After the escape points assignment, we apply A* based path searching algorithm to escape the signal to the on-grid boundary. We define a rip-up bound ψ which is a maximum rip-up net number for a searching, it will be decreased at each searching iteration. In this work, each un-route or ripped up net will perform an A* based escape path searching. At each iteration, each net could rip-up blocking nets within valid number, those net would be re-routed at next iteration.

After ψ becomes 0, there may have some nets which still fail to escape out of the boundary. For these nets, we redefine the termination condition of searching, then perform A*-search again. The search will be terminated when it reaches any point outside boundary. Meanwhile, in this phase, we do not allow any net to be ripped up.

The detailed A* based escape path searching algorithm is shown in Algorithm 1, and we use the cost definition similar to [8]. The cost from node n_s to its neighbor n_a on the SRG, denoted as $COST(n_s, n_a)$, is defined as follows:

$$COST(n_s,n_a) = (p+h) \times s + 2\sum_i s_i \times (\Delta p + h) \quad (2)$$

where p is the distance from n_a to start point(bump pad), h is the distance from n_a to target(escape point), s is the spacing of current net, i is ith net blocks the current node to n_a, s_i



(a) Path searching for net B (b) Path tracing with tree structure

Fig. 3. Path finding for escape routing. The path 1 of net B has lower cost compared to path 2. A* based searching prefers to visit the path with lower cost, so path 1 is traversed first. However, the path 1 is blocked by net A, the path 2 is a better choice without ripping up any net.

is the spacing of i, the spacing of net i is s_i and Δp is the distance from n_s to n_a.

This work uses a tree structure to record each path to prevent any path passing a node twice. If we use visit flag to determine the passing between two nodes is valid or not, it will help shrink the solution space. As shown in Fig. 3(a), A* search would first traverse the lower cost path (path 1), then set the nodes on path 1 to be visited. However, the path 2 is a better solution in this case, it can not be found because some nodes of path 2 were already visited.

As shown in Fig. 3(b), in this structure, each path from root to leaf represents a searching path. Besides, the tree recording forbids any parent node to be duplicate of its child, that is, any path cannot visit a node twice. By using tree structure to record visited state of a node, we can record each path. After finding a target, we can trace the path back to the root to obtain a searching path. In this way, we can effectively find all possible paths when using Algorithm 1 in our escape routing.

Algorithm 1: A* based escape path search

Input: Each net net_i, substrate routing graph SRG, maximum capacity of edge C_{max}
Output: Escape routing result

Create a heap H;
Calculate all neighbors cost of start point;
Push these neighbors into H;
$Target_{end} = Null$;
while *H is not empty* **do**
  Find the $point_i$ with minimal cost in heap H;
  Delete $point_i$ from the heap H;
  if *# Rip up net of* $point_i$ *> bound* **then**
    continue;
  end
  if $point_i$ *is the escape target* **then**
    $Target_{end} = point_i$;
    break;
  end
  for *each neighbor* n_a *of* $point_i$ **do**
    if n_a *is not visited in this path* **and** *Capacity of edge*(n_i,n_a) $< C_{max}$ **then**
      Calculate COST($point_i$,n_a);
      Add n_a into path tree;
      Push n_a into H;
      if *any net* n_b *blocking path from* $point_i$ *to* p_n **then**
        Push n_b into rip-up list of n_a;
      end
    end
  end
end
if $Target_{end} \neq Null$ **then**
  Backtrace;
  Save rip-up nets of this path into rip-up list of net_i;
end

IV. Via planning

After finishing escape routing, second part is via assignment of each net to optimize wire congestion and wirelength. We apply the method in [4] with our cost definition to assign via for each net. We partition the package into four regions, and apply the via assignment method to each part. The via assignment is divided into three stages: (1) initial via assignment (2) improvement of via assignment (3) improvement with minimum wirelength on layer 2. The first stage is based on [3], so it will not be explained here. Before introducing the method, we will explain the via assignment problem, congestion evaluation and via modification in the following subsections.

A. Via assignment modeling

The escape points are labeled according to their coordinate on the boundary from left to right as $e_0, e_1, ..., e_{m-1}$, where m is the number of nets. The corresponding ball and via of net i are denoted as b_i and v_i respectively. The coordinate of v_i in an assignment is denoted as (x_i^v, y_i^v).

In this paper, the routing is monotonic routing which only allows a net passing through a horizontal line once. A monotonic routing can be obtained according to a via assignment result. The routing definition is defined as follows: A routing is a monotonic routing of a via assignment if and only if $x_i^v < x_j^v$ is satisfied for any pair of nets n_i and n_j $(i < j)$ such that $y_i^v = y_j^v$.

B. Congestion evaluation for a via assignment

The congestion evaluation accuracy is an important concern in this work. If the congestion value defined in cost function is inaccurate, it will decrease the adjustability of a via assignment. Since the number of wires passing through two via in [4] is evaluated by the difference of two via's x coordinate, the real number of passing wires may be lower than the estimated value. Furthermore, they did not consider the wire between the first candidate via array and bump pad, the wires in this region may also have high congestion problem.

To increase accuracy of congestion evaluation, we redefine the passing wire evaluation. The number of wire between v_i and v_j is $i - j + 1 - k$, where k is the number of net that its label between i and j is already connected to its via above v_i and v_j. The congestion evaluation values of via v, which are used in our cost function, are listed in Table I.

TABLE I
DEFINITIONS OF VARIABLE USED IN COST FUNCTION.

Variable	Definition
$cut_a(v)$	The number of wire above via v. If no via exists above v, $cut_a(v) = 0$
$density_l(v)$	The number of wire between via v and the left via of v over the distance of them
$density_r(v)$	The number of wire between via v and the right via of v over the distance of them
$d(v)$	The Manhattan distance from v to its ball
$F(v)$	The congestion balance of v. $F(v) = \lvert density_l(v) - density_r(v) \rvert$
$density_b(v)$	The number of wire between via v and its nearest bump point b over the distance of them

C. Improving via assignment

There are five modification methods in this paper:

- **EXC** Any two vias on the adjacent row are exchanged.
- **CEXC** Any two vias are exchanged.
- **MSEQ** Vias are moved to their adjacent grid node on a via one by one until reaching a grid node without a via.
- **ROT** Rotate vias in an unit via grid.
- **ROT(3)** Rotate only 3 vias in an unit via grid.

If we apply all modifications in via modification stage, it will seriously reduce the solution space size due to the complex move without any constraint. Therefore, we use the

modifications to keep in minimum moving manner in first improvement, like EXC and MSEQ.

The routing cost for a via assignment V, which denoted as $COST(V)$, is defined as follows:

$$COST(V) = \sum_{v \in V} \alpha cut_a(v) + \beta d(v) + \gamma F(v) + \sum_{v \in V_b} \delta density_b(v) \tag{3}$$

where α, β, γ and δ are user defined parameters, V is a via set of all net and V_b is a via set which adjacent to bump pad. The gain of a via modification V', denoted as $g_M(V')$, which is defined as $COST(V) - COST(V')$.

In our method, we apply EXC and MSEQ to modify a via assignment. The flow of via improvement is defined as follows:

- **Step 1:** Generate the initial via assignment.
- **Step 2:** Select the modification with maximum gain among all EXCs and MSEQs.
- **Step 3:** Apply the modification with its gain larger than 0 to current via assignment. Then go to Step 2 until no gain of any modification is greater than 0.

D. Further consideration on minimizing total wirelength on layer 2

In via assignment improvement, we simultaneously consider multiple factors. The result is a balance between the congestion and offset from via to its assigned ball on layer 1, so the routability on layer 2 may not be well as expected. In order to improve the routability on layer 2, we apply an improvement method to shorten the wirelength on layer 2 with acceptable congestion on layer 1.

In this stage, we apply same method in previous subsection with different gain definition, and we use the ROT, ROT(3) and CEXC for via modification. We redefine the $g_M(V')$ evaluation as follows:

$$g_M(V') = \sum_{v \in M} k + \Delta F(v) + \Delta d(v) \tag{4}$$

$$k = \begin{cases} 0 & \text{if } cut_a(v') \leq C \textbf{ and } \\ & cut_a(v) - cut_a(v') > 0 \\ cut_a(v) - cut_a(v') & \text{if } cut_a(v') \leq C \textbf{ and } \\ & cut_a(v) - cut_a(v') < 0 \\ p(C - cut_a(v')) & \text{if } cut_a(v') > C \end{cases} \tag{5}$$

where k is cut difference factor defined in equation (5), v' is the next position v to be moved, C is the maximum capacity of wire in a pitch on via grid, p is the penalty factor to reduce the assignment's congestion over C.

TABLE II

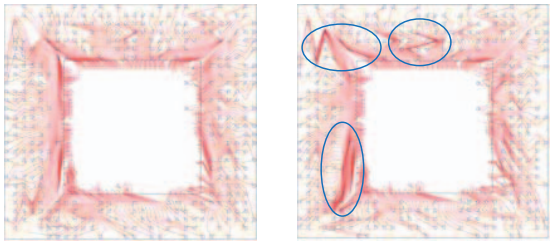
ROUTING COMPARISON BETWEEN OUR COST DEFINITION IN SEC **??** AND COST DEFINED IN [4]. WE ALSO LIST THE NUMBER OF FAIL NET WHICH DIRECTLY USE THE METHOD IN [8]. ("DEN." IS THE MAXIMUM DENSITY OF ROUTING WIRES.)

Cases	#signal	#failed net ([8])	Initial assign		[4]	Ours		
			Den.	Cost	Den.	Den.	Cost	Time (s)
Case1	340	5	21.0	2089.9	15.0	8.0	1478.3	207
Case2	379	10	46.1	2748.2	14.0	9.2	1392.8	353
Case3	380	27	18.0	2152.7	14.0	8.5	1677.7	318
Case4	387	18	25.0	2094.0	20.0	9.0	1346.7	199
Ratio	-	-	1.000	1.000	0.572	0.315	0.649	-

V. EXPERIMENTAL RESULT

We implemented proposed algorithm and flow by using C++. To evaluate our work, we applied our method on four real and large industrial cases. All of the cases contain a single chip with bumps placed in irregular positions.

Because no previous works could effectively solve such practical setup, we use the closest via assignment method in [4] for comparison. We combine their cost definition with our escape routing and via modification in all the cases we tested. Table II first lists the number of failed net routed by [8] (thus no density and cost comparision), then lists the maximum congestion and cost of each case routed by our router. Moreover, we list the result with the cost evaluation in [4] and the maximum congestion and cost from initial assignment. The illustration and comparison is shown in Fig. 4.



(a) Ours (b) With cost defined in [4]

Fig. 4. The routing prototype result of Case1. The red line is routing on top layer, the gray line is routing on bottom layer, the blue circles are assigned vias of each net, all the squares (tiny dots inside the inner rectangle) are the bumps and the orange circles are the balls. (a) is the result derived by cost defined in this paper and (b) is derived by using the cost defined in [4]. And the congested wires are shown in the blue circles.

After applying our method, we can obtain a routing prototype with less congestion on top layer and minimized wirelength on layer 2. Although some of the congested wire may violate the spacing constraint, most wires between vias are in an acceptable number. After applying our method, the package designer can observe whether an assignment between bumps and balls is acceptable in few minutes.

VI. CONCLUSION

We introduce a framework to handle pre-assignment substrate routing with irregular bump placement. First we use escape routing to handle irregular bumps, and we apply a via assignment method to improve the congested wires. Besides, the proposed path tracing based on tree structure can effectively find a path on SRG with multi-capacity edges. Also, the modified cost function can improve the adjustability of a via assignment. Through our method, it could significantly shorten the back-and-forth process in package design.

REFERENCES

[1] J.-W. Fang, C.-H. Hsu, and Y.-W. Chang. An integer-linear-programming-based routing algorithm for flip-chip designs. *IEEE TCAD*, 28(1):98–110, 2009.

[2] J.-W. Fang, M. D. Wong, and Y.-W. Chang. Flip-chip routing with unified area-I/O pad assignments for package-board co-design. In *Proc. of ACM/IEEE DAC*, pages 336–339, 2009.

[3] Y. Kubo and A. Takahashi. Global routing by iterative improvements for two-layer ball grid array packages. *IEEE TCAD*, 25(4):725–733, 2006.

[4] Y. Kubo and A. Takahashi. Routability driven modification method of monotonic via assignment for 2-layer ball grid array packages. *Proc. of IEEE/ACM ASP-DAC*, pages 238–243, 2008.

[5] B.-Q. Lin, T.-C. Lin, and Y.-W. Chang. Redistribution layer routing for integrated fan-out wafer-level chip-scale packages. In *Proc. of IEEE/ACM ICCAD*, pages 1–8, 2016.

[6] C.-W. Lin, P.-W. Lee, Y.-W. Chang, C.-F. Shen, and W.-C. Tseng. An efficient pre-assignment routing algorithm for flip-chip designs. *IEEE TCAD*, 31(6):878–889, 2012.

[7] T.-C. Lin, C.-C. Chi, and Y.-W. Chang. Redistribution layer routing for wafer-level integrated fan-out package-on-packages. In *Proc. of IEEE/ACM ICCAD*, pages 561–568, 2017.

[8] S. Liu, G. Chen, T. T. Jing, L. He, T. Zhang, R. Dutta, and X.-L. Hong. Substrate topological routing for high-density packages. *IEEE TCAD*, 28(10):207–216, 2009.

[9] T.-C. Yu and S.-Y. Fang. Flip-chip routing with IO planning considering practical pad assignment constraints. In *Proc. of IEEE/ACM ASP-DAC*, pages 521–526, 2018.