Fast Monotonic Via Assignment Excluding Mold Gates for 2-Layer Ball Grid Array Packages

Yoichi TOMIOKA[†] Atsushi TAKAHASHI[†]
[†] Department of Communications and Integrated Systems
Tokyo Institute of Technology
2–12–1–S3-58 Ookayama, Meguro–ku, Tokyo 152-8552, Japan
{yoichi,atsushi}@lab.ss.titech.ac.jp

Abstract— Ball Grid Array packages in which I/O pins are arranged in a grid array pattern realize a number of connections between chips and a printed circuit board, but it takes much time in manual routing. We propose a fast routing method for 2-layer Ball Grid Array packages to support designers. Our method obtains a via assignment which distributes wires evenly on top layer and has high completion ratio of nets by improving via assignment iteratively.

I. Introduction

In current VLSI circuits, there can be hundreds of required I/O pins. Instead of Dual In-line Package (DIP) or Quad Flat Package (QFP), in which the number of available I/O pins is small, Ball Grid Array (BGA) packages are used to realize the huge number of connections between VLSI chips and printed circuit boards (PCBs).

Since the structure of a basic BGA package is symmetrical, symmetrical radial wiring can be applied if a netlist is not specified. In many instances, however, a netlist is given and this may change frequently though positions of most pads are usually fixed in advance because package design is processed concurrently with chip design and PCB design or follows them.

The first approach for BGA package routing was proposed in [1] and was improved in [2]. In these approaches, it is assumed that there is a single routing layer, and that a netlist is not given. These approaches generate a netlist and a global route for each net. Each net connects a finger and a ball. The objective is to balance the congestion over the routing area and shorten the wire length of each net. Since a netlist is usually given in package routing design, these approaches are primarily used for package architecture design or flip-chip bonding design. For a given netlist in two-layer BGA model, as shown in Figure 1, global routing on layer 1 may be possible by using these algorithms if a candidate of via positions is considered as a ball. The feasibility of the global routes on layer 2, however, is not guaranteed.

Algorithms for multi-layer Pin Grid Array (PGA) and BGA routing have been proposed in [3] and [4], respectively. These algorithms first assign each net to a layer and then generate routes in each layer. However, neither the feasibility of the routes from the finger of each net to the assigned layer nor the routes from the assigned layer to the ball of the net are guaranteed. These routes require vias that are large compared to the wire width, and these algorithms omit the via assignment planning, which is the most difficult part of package routing.

A via assignment and global routing method for single-chip two-layer BGA packages that considers total wire length and wire congestion have been proposed as the first stage of package substrate routing in [5], and this method has been improved in [6]. In these papers, the concepts of monotonic global routing and monotonic via assignment focusing mainly on layer 1 are introduced. In the method, a via assignment is iteratively improved to minimize the maximum wire congestion on layer 1 while the total wire length on layer 2 is kept to be small enough.

Though the methods in [5] and [6] achieves small total wire length and congestion, several enhancements are required in order to use the method in actual package routing design. In the method, since the via of a net is placed near the ball of the net, the wire of each net on layer 2 is short and the routing on layer 2 seems not to be difficult. However, there is no guarantee that 100% routing on layer 2 is possible. Moreover, in package substrate, various kinds of obstacles exist. For example, mold gates from which resin is poured into the package are placed on layer 1. In the region at which a mold gate placed, routing on layer 1 is not allowed but routing on layer 2 is allowed. Mold gates make it difficult to generate 100% routing since the via of a net may be placed away from its ball if the ball is under a mold gate. Even if the evaluation of a via assignment by cost function is better, it can not be adopted if 100% routing is impossible.

In this paper, we propose a fast via assignment method which is an enhancement of the method proposed in [6] and which takes the existence of mold gates into account. In our proposed method, the maximum wire congestion on layer 1 and the wire length of a net on layer 1 and layer 2 are minimized. Though our via assignment method is based on the via assignment modification proposed in [6], the computational complexity to obtain the maximum gain is improved from $O(N^2)$ to $O(N)$, where N is the number of grid nodes. Though a via assignment obtained by fast via assignment may not realize all routes on layer 2, it is expected that the high completion ratio of nets is achieved since the total wire length on layer 2 is kept to be small

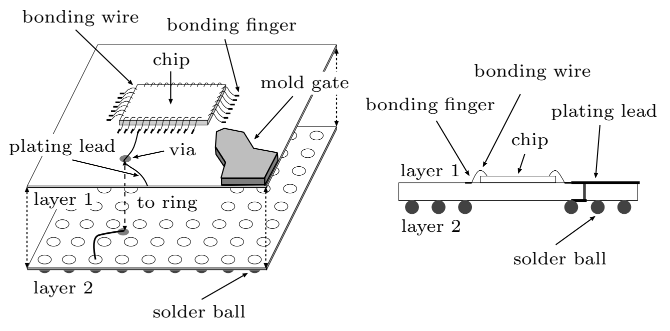

Fig. 1. A model of 2-layer BGA package.

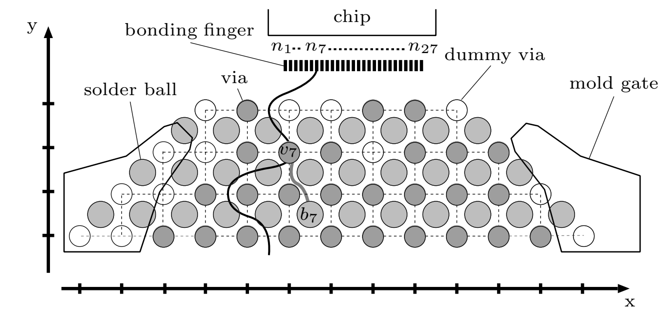

Fig. 2. Bottom sector.

enough. In order to confirm the routability of layer 2, a routing graph which corresponds to routing region on layer 2 is introduced, and routing on layer 2 is realized on it.

II. Preliminary

A. Problem definition

In this paper, we consider a basic model of BGA package as shown in Figure 1. Our BGA package model has two routing layers and single chip which is smaller than package size. A bonding finger, which we will refer to as a finger, is connected to the chip by a bonding wire. Bonding fingers are placed on the perimeter of a rectangle enclosing the chip on layer 1. A solder ball, which we will refer to as a ball, is an I/O pin of the package, and is connected to the PCB. Solder balls are placed in a grid array pattern on layer 2. There are connection requirements between bounding fingers and solder balls. The connection requirement is called a net, and is realized by wires on each layer and vias which connect wires on different layer. The number of vias to be placed in the area surrounded by four adjacent balls is at most 1. Mold gates are in some corners of top layer to pour resin into the package. In the region at which a mold gate placed, routing on layer 1 is not allowed but routing on layer 2 is allowed. Also, a via is not allowed in the region at which a mold gate placed.

Ring structure which is used for electric plating surrounds the package. Each net should be connected to the ring in order to enable electric plating to protect its wires. The extra connection to the ring of a net is called a plating lead. The ring is cut when the package is used. A plating lead is redundant for operation, but is normally used to reduce the fabrication cost and to improve the reliability.

The routing area of a package is usually divided into sectors. Our approach is applied to each sector. In the following, we focus on the bottom sector as shown in Figure 2.

In this paper, we assume that a net consists of a finger and a ball. Nets are labeled according to the order of fingers on the perimeter from the left to the right as $n_1, n_2, n_3, \ldots$.

Since the radius of a ball is large compared to the interval of the balls, the number of possible routes between adjacent balls on layer 2 is at most one. Therefore, routes on layer 2 should be short and most of plating leads should be routed on layer 1. For this reason, we restrict the route of each net so that it has only one via, the wire on layer 1 connects the finger of the net and the ring through the via of the net, and the wire on layer 2 connects the ball and the via of the net as in [5] and [6].

The set of candidate locations of vias which include the locations within a mold gate is represented by the via grid array $\mathbf{N}$. The interval of via grid array $\mathbf{N}$ is the same as that of the balls, and is unit length as shown in Figure 2. An element in $\mathbf{N}$ is called a grid node. We assume the number of possible routes on layer 1 between two vias placed in adjacent grid nodes under a design rule is at most h where h is the number of rows of balls.

The ball and the via of net n_i are denoted by b_i and v_i, respectively. The positions of b_i and v_i are denoted by (x_i^b, y_i^b) and (x_i^v, y_i^v), respectively.

Let $\mathbf{V}$ be the set of vias of the nets. A via assignment to $\mathbf{N}$ is represented by bijection $\Phi : \mathbf{V} \cup \mathbf{E} \rightarrow \mathbf{N}$, where $\mathbf{E}$ is the set of dummy vias.

The routing problem for a two-layer BGA package is defined as follows:

A routing problem for 2-layer BGA

Input: Fingers, balls, and netlist (Connection Requirements between fingers and balls)

Output: A via assignment Φ, corresponding routing on layer 1, and routing on layer 2

Objective: Minimize total wire length and maximum wire congestion

Constraint: All nets are realized, and vias are placed out of mold gates.

B. Monotonic via assignment

If the route of each net on layer 1 from its finger to the outer ring intersects every horizontal grid line only once, then the route is said to be monotonic. Otherwise, it is said to be non-monotonic. It is clear that a monotonic routing is possible for via assignment Φ 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 $y_i^v = y_j^v$. A via assignment is said to be monotonic if a monotonic routing of layer 1 is possible without considering the wire congestion [5, 6].

Given a monotonic via assignment, monotonic routing on layer 1 is uniquely determined. The via assignment shown in Figure 3 is monotonic, and its routing is unique. For ex-

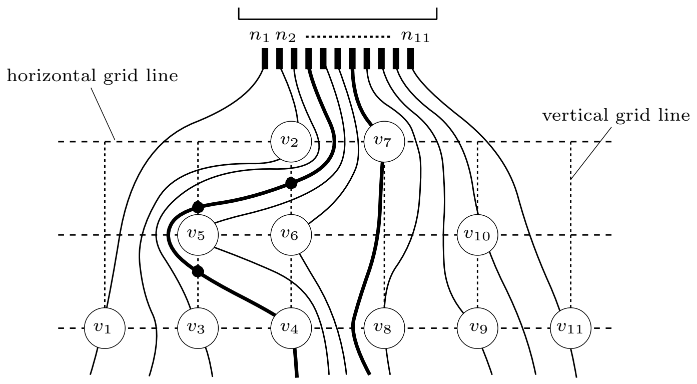

Fig. 3. A monotonic routing corresponding to a monotonic via assignment.

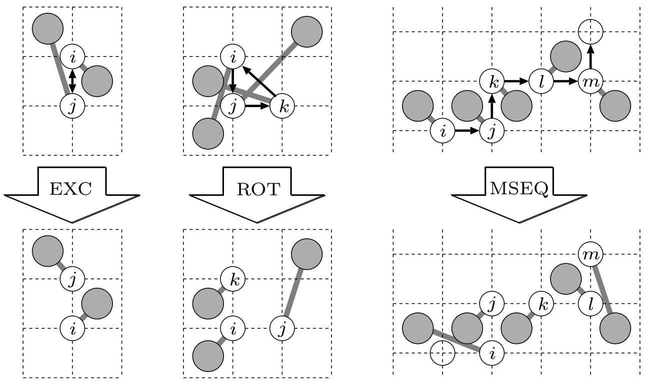

Fig. 4. Examples of three modifications.

ample, three vias v_5, v_6 and v_{10} are assigned on the middle row in Figure 3. The route of nets n_1, n_2, n_3, and n_4 in monotonic routing need to pass to the left of v_5 as shown in Figure 3.

C. *Evaluation of a via assignment*

The wire length and wire congestion are targets in the optimization of a via assignment as in [6]. In this section, indices of a via assignment which are used in the evaluation of the via assignment is explained briefly.

The number of wires on layer 1 between v and the via above v is denoted by $cut_a(v)$. If no via exists above v, $cut_a(v)$ is zero. Details are explained in [6]. The Manhattan distance between via v and the ball of the net is denoted by $d(v)$.

The wire congestion of layer 1 between v and the via to the left of v is denoted by $density_l(v)$. That is, $density_l(v)$ is the number of wires of layer 1 between them over the distance between them. If no via exists to the left of v and v is in the routing region, then $density_l(v)$ is the wire congestion of layer 1 between v and the left boundary of routing region. If no via exists to the left of v and v is within the mold gates, then $density_l(v)$ is the number of wires of layer 1 which pass to the left of v over the distance between v and the left boundary of the sector. Similarly, $density_r(v)$ is defined.

The balance of wire congestion of v is denoted by $F(v)$. That is, $F(v) = |density_l(v) - density_r(v)|$.

The illegality of via v is denoted by $obs(v)$. That is, if v is on a mold gate, $obs(v) = 1$. Otherwise, $obs(v) = 0$. $obs(v)$ is used to move vias out of a mold gate.

D. *Modifications*

There are many ways to modify a via assignment. In [6], three simple modifications are proposed which are listed below.

(EXC) Two adjacent vias on a vertical grid line are exchanged

(ROT) Three vias on a unit square on the via grid array are rotated

(MSEQ) Vias are moved to their adjacent grid nodes on a via grid array one by one until reaching a grid node without a via in which the direction of every horizontal movement of vias is either left or right and that of every vertical movement is either above or below.

Examples of above three modifications is shown in Figure 4.

III. Outline of our method

Fast Monotonic Via Assignment (FMVA) : Firstly, an initial monotonic via assignment is generated by the method proposed in [5]. Then, the initial via assignment is iteratively improved under the monotonic condition to minimize the maximum wire congestion on layer 1 while the total wire length on layer 2 is kept to be small enough.

FMVA is based on the method proposed in [6]. Three types of modification EXC, ROT, and MSEQ are used. In each iteration, a modification with the maximum gain on EXCs, ROTs, and MSEQs is applied to the current via assignment to improve the total wire length and the wire congestion. Though the initial via assignment has vias placed on a obstacle, all vias are moved to routing region in this iterative modification. In [6], it takes $O(|\mathbf{N}|^2)$ time to find an MSEQ with the maximum gain. However, we will show that it can be obtained in $O(|\mathbf{N}|)$. Therefore, each iteration takes only $O(|\mathbf{N}|)$ time.

After a via assignment is obtained by FMVA routing on layer 2 is realized. The routing graph corresponding to a routing problem on layer 2 is introduced, and routes are generated on it.

IV. Fast monotonic via assignment method

A. *Cost of a via assignment*

The routing cost for monotonic via assignment used in [6] is extended to move vias out of mold gate since the initial via assignment may have vias on a mold gate.

The routing cost for monotonic via assignment Φ, which is denoted by $COST_1(\Phi)$.

$cut_a(v)$, $d(v)$, $F(v)$, and $obs(v)$ are the number of wires on layer 1 between via v and the via above v, the Manhattan distance between via v and the ball of the net, the balance of wire congestion, and the illegality of via, respectively. They are included in the routing cost, and it is defined as follows:

$$COST_1(\Phi) = \sum_{v \in \mathbf{V}} (\alpha_1 cut_a(v) + \beta_1 d(v) + \gamma_1 F(v) + \delta_1 obs(v))$$

where α_1, β_1, γ_1, and δ_1 are coefficients. Note that δ_1 is set to much lager than the others in order to obtain a feasible via assignment.

B. The maximum gain computation

In our method, a modification with the maximum gain under the monotonic condition is selected and applied to the current via assignment. The number of patterns on EXCs and ROTs is $O(|\mathbf{N}|)$, which is small enough to enumerate all the patterns, while the number of patterns on MSEQs is exponential in the terms of the number of grid nodes. In order to find a MSEQ with the maximum gain in polynomial time, the cost graphs are used.

A cost graph is a directed acyclic graph (DAG), and has some sources and sinks. All sources in the graph correspond to the start vias of MSEQs, and all sinks in the graph correspond to the end dummy vias of MSEQs. Every directed path from source to sink corresponds to a MSEQ, and the length of the path corresponds to the gain on the MSEQ. The type of an MSEQ is either above-left, above-right, below-left, or below-right since the directions are restricted. An MSEQ with the maximum gain is obtained by generating cost graphs for each type and searching a longest path on the graphs.

In [6], the cost graph for every MSEQ beginning with a via is constructed and a longest path in each cost graph is obtained. A longest path of DAG can be obtained in $O(n+m)$, where n and m are the numbers of vertices and edges, respectively. Since the numbers of vertices and edges in a cost graph for each via are $O(|\mathbf{N}|)$, a longest path is obtained in $O(|\mathbf{N}|)$ for each cost graph. The maximum gain on MSEQs can be obtained in $O(|\mathbf{N}|^2)$ since the number of cost graphs is $O(|\mathbf{N}|)$.

In the following, we show that the cost graphs beginning with different vias can be combined. Since just four cost graphs are constructed where the numbers of vertices and edges are $O(|\mathbf{N}|)$, a MSEQ with maximum gain can be obtained in $O(|\mathbf{N}|)$.

An MSEQ M is represented by a sequence of vias, where the last via is dummy.

Let $g(M)$ be the gain of an MSEQ M that is defined by $COST(\Phi) - COST(\Phi')$, where Φ' is the via assignment obtained from Φ by applying M. For any MSEQ M, $g(M)$ can be represented as the sum of local gains $g_M(v)$, where v is a via contained in M. $g_M(v)$ can be calculated if the subsequence of M which consists of four vias around v is known, as described in [5]. Namely, even if two MSEQs M_1 and M_2 begin with different vias, $g_{M_1}(v)$ and $g_{M_2}(v)$ are the same if the subsequences of M_1 and M_2 around v are the same.

```
ConstructCostGraph(A via assignment $\Phi$)
  $\mathbf{X} \leftarrow \mathbf{V} \cup \mathbf{E}$
  $\mathbf{C} \leftarrow$ the set of vias
        s.t. no via in $\mathbf{X}$ exists in above-right of them
  while $\mathbf{C} \neq \emptyset$ do
    select $v$ from $\mathbf{C}$
    Let $v_n$ be the via above or to the right of $v$
    Let $v_p$ be the via lower or to the left of $v$
    Let $v_{pp}$ be the via lower or to the left of $v_p$
    if $v$ is dummy then
      generate feasible vertices $(\emptyset, v_p, v)$ and $(v_{pp}, v_p, v)$
    else
      if $(\emptyset, v, v_n)$ exists then
        generate $(\emptyset, \emptyset, v)$
        generate an edge from $(\emptyset, \emptyset, v)$ to $(\emptyset, v, v_n)$
      if $(v_p, v, v_n)$ exists then
        generate feasible vertices $(\emptyset, v_p, v)$ and $(v_{pp}, v_p, v)$
        generate edges from these vertices to $(v_p, v, v_n)$
    $\mathbf{X} \leftarrow \mathbf{X} \backslash \{v\}$
    $\mathbf{C} \leftarrow$ the set of vias
        s.t. no via in $\mathbf{X}$ exists in above-right of them
  done
```

Fig. 5. Algorithm of cost graph construction for above-right direction.

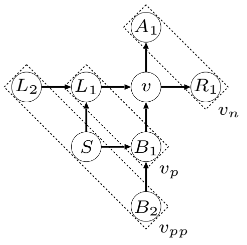

Fig. 6. Vias used to calculate $g_M(v)$.

Each vertex of a cost graph is labeled by the sequence of three vias. In a cost graph, a subsequence (v_{pp}, v_p, v, v_n) of M corresponds to vertex (v_{pp}, v_p, v), vertex (v_p, v, v_n), and the edge between them with weight $g_M(v)$, where v_n is the next via of v in M, v_p is the previous via of v in M, and v_{pp} is the previous via of v_p in M. Note that only v_n is allowed to be dummy.

The algorithm of the above-right type cost graph construction is shown in Figure 5. In a cost graph, the number of vertices in which v is the last element of label is at most seven. Note that a vertex is not generated if a via assignment becomes non-monotonic when the modification corresponding to the vertex is applied. The number of edges incident from a vertex is at most two. Therefore, the numbers of vertices and edges of a cost graph are $O(|\mathbf{N}|)$. For example, in the via assignment shown in Figure 6, labels in which v is the last element are (L_2, L_1, v), (S, L_1, v), (S, B_1, v), (B_2, B_1, v), $(\emptyset, L_1, v)$, $(\emptyset, B_1, v)$, and $(\emptyset, \emptyset, v)$. The edges incident from (L_2, L_1, v) are (L_1, v, A_1) and (L_1, v, R_1). Therefore, a modification with the maximum gain on MSEQs can be obtained in $O(|\mathbf{N}|)$.

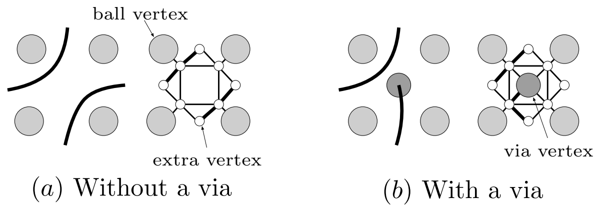

Fig. 7. Routing subgraphs on layer 2.

TABLE I
The initial cost.

	#net	C	D	F	OBS.	C+D+F
data1	316	255	316	398.72	21	969.72
data2	192	447	192	706.00	16	1345.00
data3	160	370	168	503.55	20	1041.55
data4	160	353	164	471.42	16	988.42
data5	160	276	160	457.07	14	893.07

V. Routing on layer 2

The routing graph representing routing resource on layer 2 is constructed. The structure of it is changed depending on a via assignment. The routing graph has ball vertices, via vertices, and extra vertices. A ball vertex and a via vertex correspond to a ball and a via, respectively. The number of routes intersecting between two adjacent balls is at most one since ball radius is so big. A subgraph of a routing graph in Figure 7(a) corresponds to a grid in which a ball exists in each corner and to which a via is not assigned. A subgraph of a routing graph in Figure 7(b) corresponds to a grid to which a via is assigned.

A global routing on layer 2 is obtained by using a rip-up and reroute technique on the routing graph. A shortest path of each net is sequentially generated on the routing graph regarding the routes of the other nets as obstacles. If the route of a net can not be found, then a shortest path is generated in the graph without other routes, and the routes of the other nets which intersect the found shortest path are ripped up. Whenever a route is ripped up, the weight of each vertex on the ripped up route and on the found shortest path is increased to avoid iterations such as the routes of two nets are alternately generated and ripped up.

VI. Experiments and Results

We implemented the proposed method in the C++ language and applied it to several test cases having mold gates as shown in Figure 2. The number of rows of balls is 4 in all data. The program ran on a personal computer with a 3.4GHz CPU and 1 GB of memory.

In our experiment, the number of used edges on routing graph is used as the total wire length of routing on layer 2. α_1, β_1, and γ_1 are set to 1. δ_1 is set to much larger than the others.

In the tables, C, D, F, and OBS. are $\sum cut_a(v)$, $\sum d(v)$, $\sum F(v)$, and $\sum obs(v)$, respectively, and C+D+F is the sum of them.

Let MAX be the allowable congestion of layer 1 to satisfy the design rule. MAX is depending on h. If $density_l(v_i) \leq MAX$, then the violation between v_i and the via to the left of v_i is 0. Otherwise, the violation is $density_l(v_i) - MAX$. Let Δ be the sum of the violations for a whole via grid. U is the number of unconnected nets for routing on layer 2.

In order to confirm the speed-up of gain computation, the gain computation proposed by [6] is also implemented, and is denoted by OLD in TABLE II.

The initial cost and the result of each data are shown in Table I and II, respectively. All data is divided into four sectors, and each method is applied to them.

The final solution for most inputs is improved drastically as shown in Table II. Although the method proposed in [6] needs 45 second for data1, our proposed method obtains the identical output within 3 second due to improvement of the computational complexity of each iteration.

The initial routes for data4 is shown in Figure 8, and the final solution is shown in Figure 9. Unconnected nets are connected by straight line in Figure 9. Mold gates are not drawn in these figures.

Though our method does not realize all net, the number of unconnected nets is small. Therefore, it is expected that all nets can be realized if small modification is applied to the obtained via assignment in the post-processing. But, this is in our future works.

In addition, though most wires are distributed evenly, there exist places with high wire congestion near mold gates. This is caused by that moving vias out of mold gates has priority over improving or maintaining the wire congestion and the distance between a via and a ball in the first phase. These bad effects will be relaxed if the initial via assignment in which vias are placed out of a mold gate is created with the routability analysis, and the wire congestion on layer 1 can be improved if parts of plating leads is realized on layer 2. But, these are in our future works.

VII. Conclusion

We proposed a monotonic via assignment method which is an enhancement of the method based on [6] and which takes the existence of mold gates into account. We showed that a modification with the maximum gain is obtained in $O(|\mathbf{N}|)$, though it takes $O(|\mathbf{N}|^2)$ times in [6]. Moreover, we gave a routing graph for routing on layer 2, and routing design on both layers is generated.

In our experiments, our method obtains a via assignment which distributes wires evenly faster than the method proposed in [6], and most of nets are realized for the assignment. Our proposed method explores monotonic via assignments effectively.

In our future work, we will propose the method to create the initial via assignment in which vias are placed out of a mold gate with the routability analysis. Moreover, we will consider how to apply modifications to improve routability.

TABLE II
The result of the first modification

	Final Solution						Violation		Fast Monotonic Via Assignment					
	COST								#Modification				Exe. time [sec]	
	C	D	F	OBS.	C+D+F	(imp.)	Δ	U	EXC	ROT	MSEQ	ALL	OLD	FMVA
data1	118	412	233.98	0	763.98	(21.2%)	2.80	11	11	0	48	59	44.96	2.58
data2	147	284	349.67	0	780.67	(41.9%)	47.6	3	20	5	32	57	8.37	0.86
data3	81	255	165.52	0	501.52	(51.8%)	7.28	5	16	5	57	78	10.58	1.29
data4	63	241	171.40	0	475.40	(51.9%)	11.5	3	21	2	45	68	9.57	1.09
data5	68	216	153.45	0	437.45	(51.0%)	7.14	0	12	2	41	55	8.33	0.85

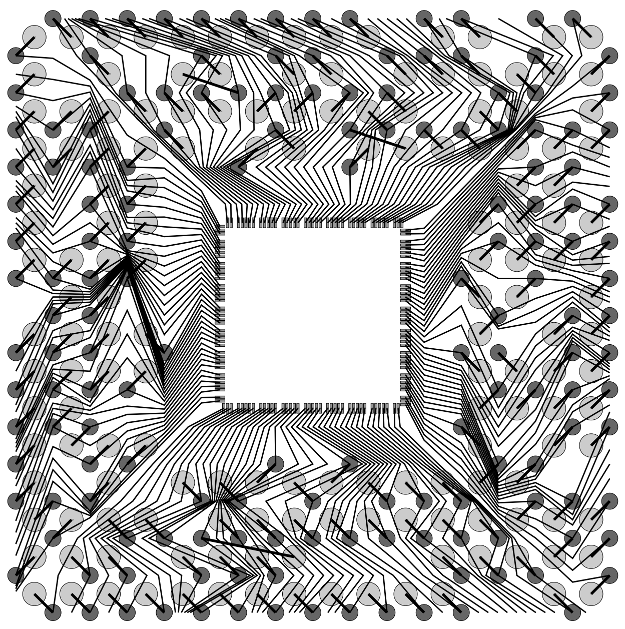
Fig. 8. The initial routes on layer 1 for data4.

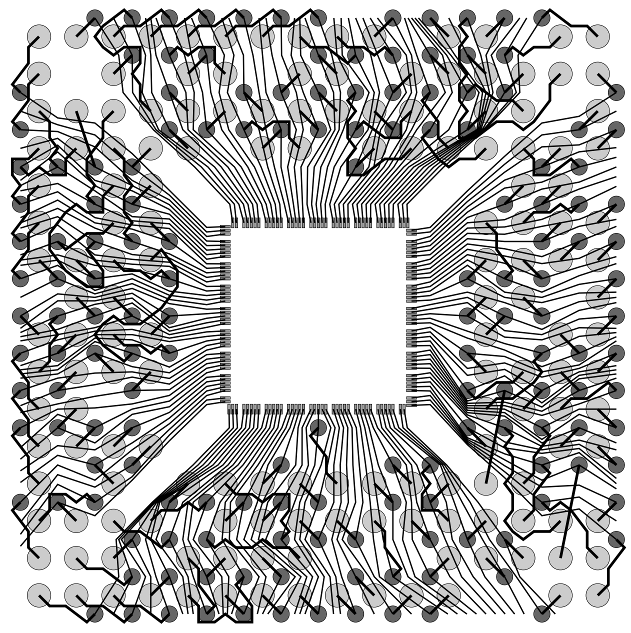
Fig. 9. The output routes for data4.

Acknowledgements

This research was partially supported by Grant-in-Aid for Scientific Research (C) (18500034) and Grant-in-Aid for JSPS Fellows (19·9340).

References

[1] M.-F. Yu and W. W.-M. Dai, "Single-Layer Fanout Routing and Routability Analysis for Ball Grid Arrays," in *Proceedings of International Conference Computer-Aided Design*, pp. 581–586, 1995.

[2] S. Shibata, K. Ukai, N. Togawa, M. Sato, and T. Ohtsuki, "A BGA Package Routing Algorithm on Sketch Layout System," *The journal of Japan Institute for Interconnecting and Packaging Electronic Circuits*, vol. 12, no. 4, pp. 241–246, 1997. (In Japanese).

[3] C.-C. Tsai, C.-M. Wang, and S.-J. Chen, "NEWS: A Net-Even-Wiring System for the Routing on a Multilayer PGA Package," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 2, pp. 182–189, 1998.

[4] S.-S. Chen, J.-J. Chen, C.-C. Tsai, and S.-J. Chen, "An Even Wiring Approach to the Ball Grid Array Package Routing," in *Proceedings of International Conference on Computer Design*, pp. 303–306, 1999.

[5] Y. Kubo and A. Takahashi, "A Via Assignment and Global Routing Method for 2-Layer Ball Grid Array Packages," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88-A, no. 5, pp. 1283–1289, 2005.

[6] Y. Kubo and A. Takahashi, "Global Routing by Iterative Improvements for 2-Layer Ball Grid Array Packages," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, no. 4, pp. 725–733, 2006.