Article Information

Title	Routing of Monotonic Parallel and Orthogonal Netlists for Single-Layer Ball Grid Array Packages
Authors	Yoichi TOMIOKA and Atsushi TAKAHASHI
Citation	IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E89-A, No.12, pp.3551-3559
Copyright	
IEICE Transactions Online URL	https://search.ieice.org/

PAPER Special Section on VLSI Design and CAD Algorithms

Routing of Monotonic Parallel and Orthogonal Netlists for Single-Layer Ball Grid Array Packages

Yoichi TOMIOKA[†a)], *Nonmember and* **Atsushi TAKAHASHI**[†b)], *Member*

SUMMARY Ball Grid Array packages in which I/O pins are arranged in a grid array pattern realize a number of connections between chips and PCB, but it takes much time in manual routing. So the demand for automation of package routing is increasing. In this paper, we give the necessary and sufficient condition that all nets can be connected by monotonic routes when a net consists of a finger and a ball and fingers are on the two parallel boundaries of the Ball Grid Array package, and propose a monotonic routing method based on this condition. Moreover, we give a necessary condition and a sufficient condition when fingers are on the two orthogonal boundaries, and propose a monotonic routing method based on the necessary condition.

key words: *ball grid array, monotonic, single-layer, package, routing*

1. Introduction

Ball Grid Array (BGA) packages as shown in Fig. 1, in which I/O pins are placed in a grid array pattern, realize a number of connections between chips and the printed circuit board (PCB). Bonding fingers are connected to chips, and solder balls are I/O pins of the package in a grid array pattern. Since the structure of BGA packages is simple, many routes can be realized in few layers in the packages if connection requirements and routing patterns are suitable for the structure. In current package routing design, the designer generates satisfactory routing patterns by using the properties of connection requirements effectively. But it takes much time for large packages since the huge number of routes needs to be realized. So, the demand for automation of package routing is increasing. In this paper, we consider routing for a single-layer BGA package as the first step for BGA packages routing.

In the literature on planar routing, there are a lot of problem formulations and approaches. For example, problem formulations for single-row and double-row routing, where terminals are placed on single-row and double-row, are proposed in [1] and [2], respectively. Though these problem formulations are similar to problems for single-layer BGA packages, approaches for them are not enough to obtain satisfactory routes for BGA packages. Though there exist other approaches and parts of them are included in several tools, most tools are for routing on PCBs or IC chips. Most approaches proposed so far can not immediately apply to BGA packages routing which contains special requirements and constraints on BGA packages. Actually, many parts of the routing process for BGA packages are realized manually with support tools.

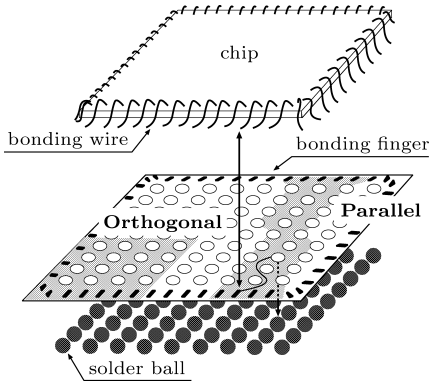


Fig. 1 Ball grid array package.

In order to obtain a satisfactory routing pattern, the analysis of manual routing patterns is necessary. We introduce know-how in manual routing into our methods to obtain satisfactory routing pattern efficiently. In manual routing patterns, though routes may snake, most of them do not go back. The routes which do not go back are said to be *monotonic*. In monotonic routing patterns, it is expected that the total wire length tends to be small, and it is easy to decide the route of each net. But there exists a netlist that cannot be realized by monotonic routes in one layer with any design rule. In cases that fingers in a netlist are on the same boundary, the necessary and sufficient condition for all nets being realized by monotonic routes is known. In this paper, we generalize it for cases that fingers in a netlist are on two boundaries.

There also exists a netlist in which a design rule may be satisfied if non-monotonic routes are allowed. In these cases, non-monotonic routes are needed. Though we aim to realize nets in one layer under a certain design rule, in this paper we propose an approach in which all nets are realized by monotonic routes. The obtained monotonic routes will be an initial solution in iterative improvement to satisfy the design rule.

In literatures for BGA package, several approaches focusing on monotonic routes were proposed. The first approach for single-layer BGA packages was proposed in [3]

Manuscript received March 13, 2006.
Manuscript revised June 13, 2006.
Final manuscript received August 1, 2006.

[†]The authors are with the Department of Communications and Integrated Systems, Tokyo Institute of Technology, Tokyo, 152-8552 Japan.
a) E-mail: yoichi@lab.ss.titech.ac.jp
b) E-mail: atsushi@lab.ss.titech.ac.jp

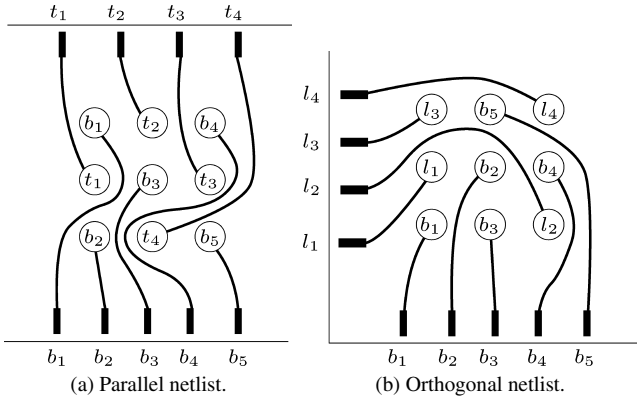
DOI: 10.1093/ietfec/e89–a.12.3551

and it was improved in [4]. Their approach generates optimal uniform distribution of wire by generating connection requirements. An approach for 2-layer BGA packages was proposed in [5]. It is given connection requirements, and optimizes the total wire length and the wire congestion by improving via assignment.

Also, several approaches considering non-monotonic routes were proposed. The approach for multilayer Pin Grid Array (PGA) and Ball Grid Array packages were proposed in [6] and [7], respectively. They assign each net to a layer, and realize nets in respective layer.

All of them divide the package into several sectors, and nets are realized within each sector. Basically, each sector consists of bonding fingers on the same boundary of the package and solder balls, and a net in each sector consists of a bonding finger and a solder ball. Namely, these approaches cannot be applied if it is impossible to divide the package into such sectors. So, we propose an approach for the region consisting of solder balls and bonding fingers on two boundaries of the package as shown in Fig. 2.

Section 2 introduces routing model, and gives some definitions for analysis. Connection requirements are the set of nets and are called a netlist. A *parallel netlist*, in which bonding fingers are on two parallel boundaries of the package, is shown in Fig. 2(a). In Sect. 3, we give the necessary and sufficient condition that all nets in the parallel netlist can be realized by monotonic routes, and propose a monotonic routing method based on this condition. An *orthogonal netlist*, in which bonding fingers are on two orthogonal boundaries of the package, is shown in Fig. 2(b). In Sect. 4, we give a necessary condition and a sufficient condition that all nets in the orthogonal netlist can be realized by monotonic routes, and propose a monotonic routing method based on the proposed necessary condition. There may exist more than one monotonic routing pattern that corresponds to a parallel netlist and an orthogonal netlist, respectively. How to select one among them that meets the design rule, if it exists, is in our future works. Since it is not guaranteed that our routing method for orthogonal netlists completes routing, we implement our method for orthogonal netlists with C++ language, and applied it to orthogonal netlists in Sect. 5. Section 6 concludes this paper.



(a) Parallel netlist. (b) Orthogonal netlist.

Fig. 2 Monotonic netlists decision problem.

2. Preliminary

2.1 Definitions

In this paper, we assume that the BGA package has connection requirements between bounding fingers placed on boundaries of the package and solder balls placed in a grid array pattern. 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. A bonding finger, which we will refer to as a finger, is connected to the chip by a bonding wire.

We assume that all nets are two-terminal nets connecting a finger to a ball. A netlist is the set of such nets and is represented by **N**, and let n be the number of nets in **N**. We refer to a finger placed on a bottom boundary of the package as a bottom finger, and refer to a net which consists of a bottom finger and a ball as a bottom net. Similarly, a top net and a left net are defined. Bottom nets and top nets are labeled according to the order of fingers from the left to the right as $b_1, b_2, b_3, \ldots$ and $t_1, t_2, t_3, \ldots$, respectively. Left nets are labeled according to the order of fingers from the bottom to the top as $l_1, l_2, l_3, \ldots$. Let **B**, **T**, and **L** be the sets of bottom, top, and left nets, respectively.

We define the relation between nets according to their ball positions as shown in Fig. 3. Let (x_a, y_a) and (x_b, y_b) be the coordinates of the balls of nets a and b, respectively. The relation between a and b is defined as follows.

- If $x_a < x_b$ and $y_a = y_b$, then a is said to be to the *left* of b and the relation is represented by ${}_aH_b$.
- If $x_a = x_b$ and $y_a < y_b$, then a is said to be *below* b and the relation is represented by ${}_aV_b$.
- If $x_a < x_b$ and $y_a < y_b$, then a is said to be to the *lower-left* of b and the relation is represented by ${}_aS^b$.
- If $x_a < x_b$ and $y_a > y_b$, then a is said to be to the *upper-left* of b and the relation is represented by ${}^aS_b$.
- ${}_bH_a$, ${}_bV_a$, ${}_bS^a$ and ${}^bS_a$ are defined symmetrically.

In addition, net a is said to be *adjacent* to net b if balls of a and b are adjacent in a row or in a column.

2.2 Order Graphs

We use some order graphs where a vertex v corresponds to

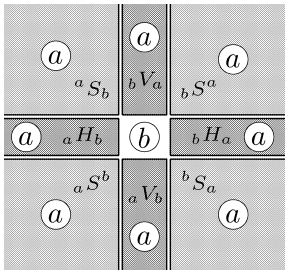
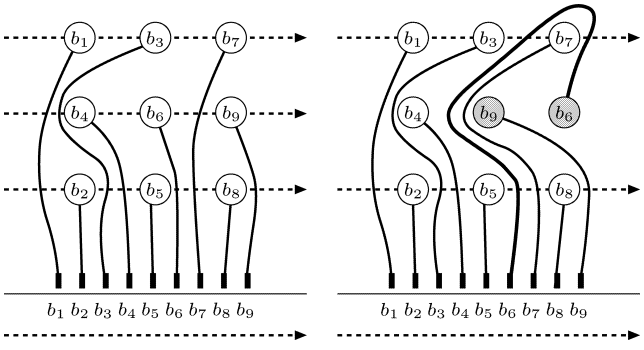
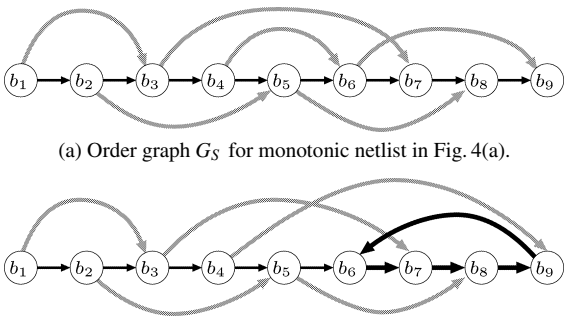


Fig. 3 Relationship between a and b.



(a) All routes are monotonic. (b) $R(b_6)$ is non-monotonic.

Fig. 4 Monotonic and non-monotonic routes.



(a) Order graph G_S for monotonic netlist in Fig. 4(a).

(b) Order graph G_S for non-monotonic netlist in Fig. 4(b).

Fig. 5 Order graphs for single netlists.

a net $v \in \mathbf{N}$. The number of vertices in each order graph is n. The edge from a vertex u to a vertex v is represented by the ordered pair (u, v). In this paper, every order graph has edges corresponding to the order of fingers in each boundary. E_f^b, E_f^t, and E_f^l are the sets of edges corresponding to the order in bottom, top, and left boundaries, respectively. Formally, they are given as follows:

$$E_f^b = \{(b_i, b_{i+1}) \mid b_i, b_{i+1} \in \mathbf{B}\},$$
$$E_f^t = \{(t_i, t_{i+1}) \mid t_i, t_{i+1} \in \mathbf{T}\},$$
$$E_f^l = \{(l_i, l_{i+1}) \mid l_i, l_{i+1} \in \mathbf{L}\}.$$

2.3 Monotonic Routes

A boundary, where the finger of a net is placed, is called the *finger boundary* of the net. For example the finger boundary of b_1 in Fig. 2(a) is the bottom boundary. In this paper, a monotonic route and a non-monotonic route are defined as follows:

Definition 1: If the route from a finger to a ball intersects any straight lines running parallel with the finger boundary at most once, then the route is said to be *monotonic*. Otherwise the route is said to be *non-monotonic*.

Let $R(v)$ be the route of a net $v \in \mathbf{N}$. All routes are monotonic in Fig. 4(a), but $R(b_6)$ is non-monotonic in Fig. 4(b).

If all nets in a netlist can be realized by monotonic routes without intersecting each other, the netlist is said to be monotonic.

A netlist is said to be *single* if fingers in the netlist are placed on the same boundary. Examples of single netlists are shown in Fig. 4. A single netlist is monotonic if and only if nets on each row are in increasing order. Since the netlist in Fig. 4(a) satisfies this condition, it is monotonic. On the other hand, the netlist in Fig. 4(b) is non-monotonic since b_6 and b_9 are in decreasing order and either $R(b_6)$ or $R(b_9)$ become non-monotonic. For a monotonic single netlist, monotonic routing pattern in which all routes are monotonic is unique. Similar observations are found in [3]–[5].

Whether a single netlist is monotonic or not is decided by the order graph G_S. Let E_S be the set of edges (a, b) where a and b are in ${}_aH_b$ and adjacent. $E(G_S)$ consists of E_f^b and E_S. For example in Fig. 4(a), G_S has edges (b_2, b_5) and (b_5, b_8) corresponding to the bottom row. Similarly, G_S has edges for other rows.

The order graph G_S corresponding to the netlist in Fig. 4(a) is acyclic (See Fig. 5(a)). But, the order graph G_S corresponding to the netlist in Fig. 4(b) is cyclic (See Fig. 5(b)). Clearly, G_S is cyclic if and only if there exist nets on a row which are in decreasing order, such as b_6 and b_9 in Fig. 4(b).

3. Parallel Netlists

A *parallel netlist* is a netlist in which fingers are placed on the two parallel boundaries of the package. In this section, we analyze parallel netlists.

3.1 Monotonic Parallel Netlists

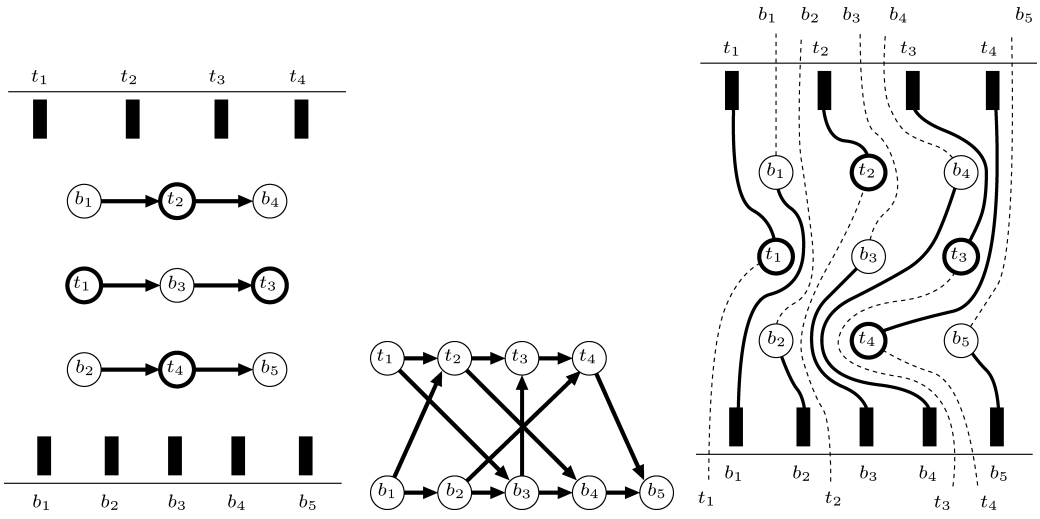
The Monotonic Parallel Netlist (MPN) Decision Problem is defined as follows:

Definition 2: MPN Decision Problem
Input:
A parallel netlist.
Question:
Is it possible to realize all connection requirements by monotonic routes?

An example of MPN Decision Problem is given in Fig. 2(a). In this case, $\mathbf{N} = \mathbf{B} \cup \mathbf{T}$. The necessary and sufficient condition for being monotonic is that nets on each row are in increasing order without distinguishing bottom and top nets. This condition is represented by the order graph G_P. Let E_p be the set of edges (x, y) where x and y are in ${}_xH_y$ and adjacent. $E(G_P)$ consists of E_f^b, E_f^t, and E_p. A parallel netlist is shown in Fig. 6(a), and its order graph G_P is shown in Fig. 6(b). In Fig. 6(a), edges in E_p are shown. A parallel netlist is monotonic if and only if G_P is acyclic.

Theorem 1: A parallel netlist is monotonic if and only if the order graph G_P is acyclic, where $E(G_P) = E_f^b \cup E_f^t \cup E_p$ and

$E_p = \{(u, v) \mid u, v \in \mathbf{N}, u \text{ is adjacent to } v, {}_uH_v\}$.



(a) Parallel netlist. (b) Order graph G_P. (c) Monotonic routing pattern.

Fig. 6 MPN decision problem.

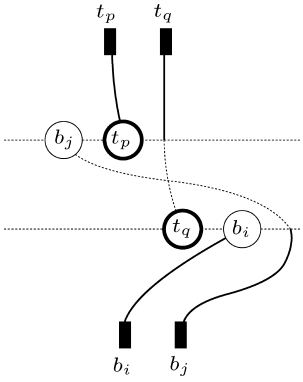


Fig. 7 ${}_{b_j}H_{t_p} \wedge {}_{t_q}H_{b_i}$.

Proof. If the order graph G_P is acyclic, then an order can be obtained by G_P. A monotonic routing pattern can be realized according to the order as we will show in Sect. 3.2. Conversely, consider that G_P has a cycle C. If C consists of only bottom nets, then non-monotonic routes are needed since it means that bottom nets are in decreasing order on a row. The same discussion is possible for top nets. So we assume that C consists of bottom nets and top nets. Without loss of generality, we assume that C has (b_j, t_p) and (t_q, b_i), where $b_i, b_j \in \mathbf{B}$ $(i < j)$ and $t_p, t_q \in \mathbf{T}$ $(p < q)$. Since ${}_{b_j}H_{t_p}$ and ${}_{t_q}H_{b_i}$, non-monotonic routes are needed by at least one of them as shown in Fig. 7. □

3.2 A Parallel Routing Method

A partial order is defined by G_P, and some orders are obtained from the partial order. This order corresponds to an order such that the sources in G_P are removed one by one. For example, the following order

$$t_1 \to b_1 \to b_2 \to t_2 \to b_3 \to b_4 \to t_3 \to t_4 \to b_5$$

is obtained from the order graph in Fig. 6(b). According to the obtained order, we put virtual fingers on bottom boundary of the package for top nets and put virtual fingers on top boundary for bottom nets. All nets can be realized because we can connect a finger to its virtual finger via its ball by monotonic route one by one from the left. An example of solution are given in Fig. 6(c).

An order is obtained from G_P in $O(n+m)$ where m is the number of edges in G_P. The number of edges is $O(n)$ since each vertex in G_P has at most two outgoing edges. Therefore, the time complexity for obtaining an order is $O(n)$.

For a monotonic parallel netlist, there are several monotonic routing patterns since an order obtained from G_P is not unique in general. Though our method in this paper selects an order obtained from G_P at random, the selection of the order that reduces the density is in our future work.

4. Orthogonal Netlists

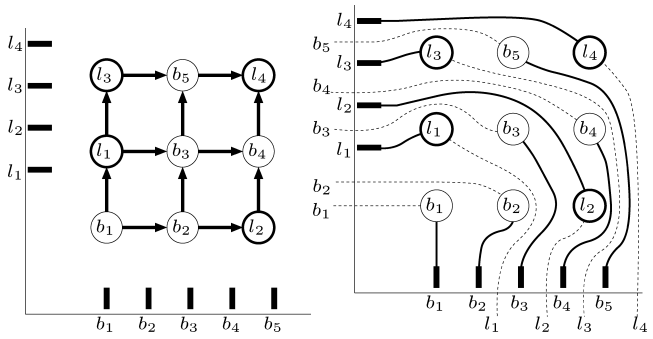
An *orthogonal netlist* is a netlist in which fingers are placed on the two orthogonal boundaries of the package. In this section, we analyze orthogonal netlists.

4.1 Monotonic Orthogonal Netlists

The Monotonic Orthogonal Netlist (MON) Decision Problem is defined as follows:

Definition 3: MON Decision Problem
Input:
An orthogonal netlist.
Question:
Is it possible to realize all connection requirements by monotonic routes?

An example of MON Decision Problem is given in Fig. 2(b). In this case, $\mathbf{N} = \mathbf{B} \cup \mathbf{L}$.



(a) Orthogonal netlist. (b) Monotonic routing pattern.

Fig. 8 MON decision problem.

4.1.1 A Sufficient Condition

An orthogonal netlist is monotonic if the order graph G_s, which has edges corresponding to the order of nets on each row and column without distinguishing bottom and left nets, is acyclic. According to the order given by G_s, we put virtual fingers. Nets are realized by connecting each finger to its virtual finger via its ball from the lower left.

Theorem 2: An orthogonal netlist is monotonic if the order graph G_s is acyclic, where $E(G_s) = E_f^b \cup E_f^l \cup E_s$ and

$E_s = \{(u, v) \mid u, v \in \mathbf{N}, u \text{ is adjacent to } v, {}_uH_v \vee {}_uV_v\}$.

An example of MON Decision Problem and a monotonic routing pattern for the netlist are given in Fig. 8. In Fig. 8(a), edges in E_s are shown.

If ball of b_2 in the second column in Fig. 8 is swapped for ball of b_3, G_s becomes cyclic. But, the netlist is monotonic since a monotonic routing pattern for the netlist is given in Fig. 2(b). So, this condition is not a necessary condition.

4.1.2 A Necessary Condition

A route can be regarded as the set of points. Let b be a bottom net and v be a net. If there exists a point (x_b, y_b) on $R(b)$ and a point (x_v, y_v) on $R(v)$ such that $x_b < x_v$ and $y_b = y_v$, then $R(v)$ is said to be to the right of $R(b)$. In other words, $R(v)$ is said to be to the right of $R(b)$ if there exists a point on $R(v)$ which is to the right of $R(b)$. Similarly, $R(v)$ for net v is said to be above $R(l)$ for left nets if there exists a point on $R(v)$ which is above $R(l)$.

Theorem 3: Let routing pattern graph G_R be the directed graph constructed for a routing pattern, where the vertices correspond to the nets, and edge set is defined as follows:

- An edge (b, v) $(b \in \mathbf{B}, v \in \mathbf{N})$ exists if and only if $R(v)$ is to the right of $R(b)$.
- An edge (l, v) $(l \in \mathbf{L}, v \in \mathbf{N})$ exists if and only if $R(v)$ is above $R(l)$.

If the routing pattern is monotonic, then G_R is acyclic.

Proof. Consider that G_R is cyclic. Let C be a cycle in G_R.

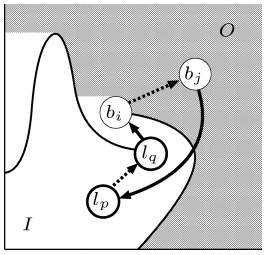


Fig. 9 A cycle in G_R.

The cycle C cannot consist of only bottom nets or only left nets since there is no non-monotonic route. So, there are edge from a bottom net to a left net and edge from a left net to a bottom net. Without loss of generality, we assume that C includes (b_j, l_p) and (l_q, b_i), and that $R(b_i)$ is above $R(l_q)$, where $b_i, b_j \in \mathbf{B}$ $(i \le j)$ and $l_p, l_q \in \mathbf{L}$ $(p \le q)$.

Let region I be the region which is to the left of $R(b_i)$ or $R(l_q)$, and region O be the other region except I. Formally,

$I = \{(x, y) \mid (x_r, y) \text{ is on } R(b_i) \text{ or } R(l_q) \text{ where } x < x_r\}$

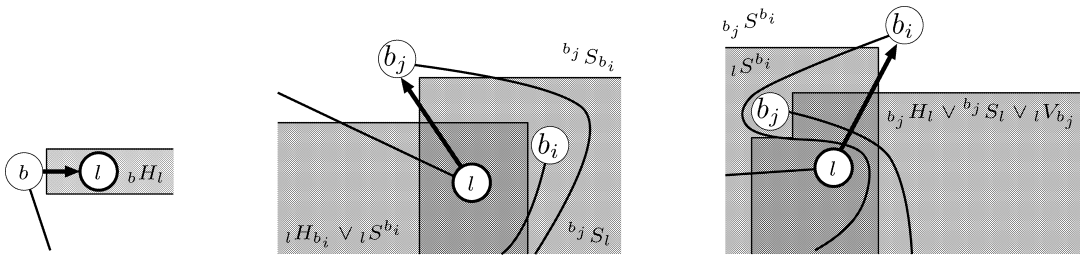
See Fig. 9. Since all routes are monotonic, the ball of b_j and $R(b_j)$ need to exist in region O shown in Fig. 9. The ball of l_p and $R(l_p)$ need to exist in region I, though region I contains region in which $R(l_p)$ can not exist if $R(l_p)$ is monotonic. Therefore, $R(l_p)$ is not to the right of $R(b_j)$, since $x_i < x_o$ if $y_i = y_o$ where (x_i, y_i) and (x_o, y_o) are points in region I and O, respectively. However, G_R has the edge (b_j, l_p). It contradicts definition of G_R. So, G_R is acyclic if all of routes are monotonic. □

G_R is not defined when a routing pattern is not given. However, depending on the relationship between bottom net b and left net l, there are cases that $R(l)$ is always to the right of $R(b)$ in any monotonic routing patterns. In such cases, edge (b, l) exists in G_R for any monotonic routing patterns. Similarly, there are cases that edge (l, b) exists in G_R for any monotonic routing patterns. The graph where only such edges exist is the graph obtained from G_R by removing some of edges. Therefore, we consider such order graph G_n. Clearly, G_n is acyclic.

For example, if a monotonic orthogonal netlist contains nets b and l of relation ${}_bH_l$ as shown in Fig. 10(a), then edge (b, l) is in G_n since $R(l)$ is always to the right of $R(b)$ in any monotonic routing patterns (Type E_h). Similarly, if a monotonic orthogonal netlist contains nets b_i, b_j, and l $(i < j)$ of relation ${}^{b_j}S_{b_i} \wedge ({}_lH_{b_i} \vee {}_lS^{b_i}) \wedge {}^{b_j}S_l$ as shown in Fig. 10(b), then edge (l, b_j) is in G_n since $R(b_j)$ is always above $R(l)$ in any monotonic routing patterns (Type E_1^b). Also, if it contains nets of relation ${}_{b_j}S^{b_i} \wedge {}_lS^{b_i} \wedge ({}_{b_j}H_l \vee {}^{b_j}S_l \vee {}_lV_{b_j})$ as shown in Fig. 10(c), then edge (l, b_i) is in G_n since $R(b_i)$ is always above $R(l)$ in any monotonic routing patterns (Type E_2^b).

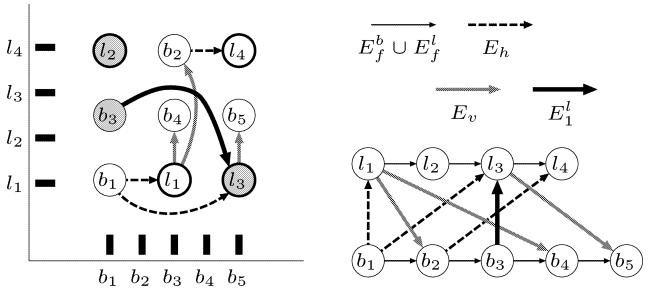
An order graph derived from necessary conditions between two or three nets is a subgraph of G_n. Let G_n' be such order graph, and an orthogonal netlist is not monotonic if G_n' is cyclic.

Theorem 4: An orthogonal netlist is not monotonic if the



(a) Type E_h. (b) Type E_1^b. (c) Type E_2^b.

Fig. 10 Examples of constraints.



(a) Orthogonal netlist. (b) Order graph G'_n.

Fig. 11 A monotonic orthogonal netlist.

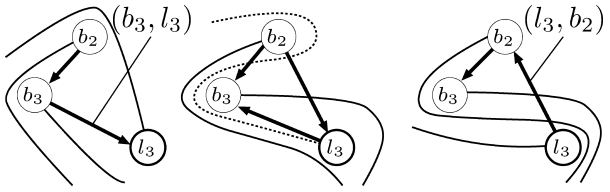


Fig. 12 Alternative constraints $(b_3, l_3) \oplus (l_3, b_2)$.

order graph G'_n is cyclic, where $E(G'_n) = E_f^b \cup E_f^l \cup E_h \cup E_v \cup E_1^b \cup E_1^l \cup E_2^b \cup E_2^l$,

$E_h = \{(b, u) \mid {}_bH_u\}$,
$E_v = \{(l, u) \mid {}_lV_u\}$,
$E_1^b = \{(l, b_j) \mid {}^{b_j}S_{b_i} \wedge ({}_lH_{b_i} \vee {}_lS^{b_i}) \wedge {}^{b_j}S_l\}$,
$E_1^l = \{(b, l_j) \mid {}^{l_i}S_{l_j} \wedge ({}_bV_{l_i} \vee {}_bS^{l_i}) \wedge {}^bS_{l_j}\}$,
$E_2^b = \{(l, b_i) \mid {}_{b_j}S^{b_i} \wedge {}_lS^{b_i} \wedge ({}_{b_j}H_l \vee {}^{b_j}S_l \vee {}_lV_{b_j})\}$,
$E_2^l = \{(b, l_i) \mid {}_{l_j}S^{l_i} \wedge {}_bS^{l_i} \wedge ({}_bH_{l_j} \vee {}^bS_{l_j} \vee {}_{l_j}V_b)\}$,

where $u \in \mathbf{N}$, $b, b_i, b_j \in \mathbf{B}$, $l, l_i, l_j \in \mathbf{L}$, and $i < j$.

G'_n in Fig. 11(b) corresponds to the orthogonal netlist in Fig. 11(a). $(b_1, l_1), (b_1, l_3)$, and (b_2, l_4) are in E_h. Also, $(l_1, b_4), (l_1, b_2)$, and (l_3, b_5) are in E_v. (b_3, l_3) is in E_1^l since there exists a condition derived from three nets b_3, l_2, and l_3.

In addition, there are alternative constraints. When three balls of nets b_2, b_3, and l_3 are placed as shown in Fig. 12, $R(l_3)$ is non-monotonic if $R(l_3)$ is to the right of $R(b_2)$ and below $R(b_3)$. So, in any monotonic routing patterns, $R(l_3)$ is either below or to the right of $R(b_2)$ and $R(b_3)$. Therefore, G_R contains either (b_3, l_3) or (l_3, b_2) for a monotonic routing pattern. This constraint is represented by $(b_3, l_3) \oplus (l_3, b_2)$.

There exists an alternative constraint $(b_j, l) \oplus (l, b_i)$ if $({}_{b_j}V_{b_i} \vee {}_{b_j}S^{b_i}) \wedge {}^{b_i}S_l \wedge {}^{b_j}S_l$, where $b_i, b_j \in \mathbf{B}$ $(i < j)$ and $l \in \mathbf{L}$. Similarly, alternative constraints for two left nets and a bottom net are defined.

Assume that there exists three alternative constraints

$$(b_2, l_1) \oplus (l_1, b_1), (b_2, l_2) \oplus (l_2, b_1), \text{ and } (b_2, l_3) \oplus (l_3, b_1).$$

All of combinations except $\{(l_1, b_1), (l_2, b_1), (l_3, b_1)\}$ and $\{(b_2, l_1), (b_2, l_2), (b_2, l_3)\}$ generate cycles in an order graph by adding their edges. Therefore, we can regard these alternative constraints as an alternative constraint $(b_2, l_1) \oplus (l_3, b_1)$. Since the number of alternative constraint corresponding to two bottom nets or two left nets is at most one, the number of alternative constraints is $O(n^2)$.

We constructed G_n^* by adding some edges that corresponds to alternative constraints to G'_n. Let the number of alternative constraints be c, and denote alternative constraints by $a_i \oplus {a'}_i$ $(1 \le i \le c)$. G_n^* is defined as follows: Let G be $G = G'_n$; If $G + a_i$ is cyclic, then $G = G + {a'}_i$; If $G + {a'}_i$ is cyclic, then $G = G + a_i$; G_n^* is obtained by repeating this operation until no edge can be added.

By the definition of G_n^*, G_n^* has all edges of alternative constraints if G_n^* is cyclic.

Theorem 5: G_n^* is well-defined.

Proof. Let G_1 and G_2 be graphs obtained by the previous procedure, and suppose that G_1 is different from G_2. let $e_1, e_2, \ldots, e_u$ and $f_1, f_2, \ldots, f_v$ be the sequences of edges added to G'_n in obtaining G_1 and G_2, respectively. G_1 has an edge that G_2 does not have. Let e_k be the first edge in the sequence $e_1, e_2, \ldots, e_u$ that is not an edge of G_2. e_k is the edge of alternative constraint $e_k \oplus {e'}_k$, and e_k was selected since $G' + {e'}_k$ is cyclic where $G' = G'_n + \{e_1, e_2, \ldots, e_{k-1}\}$. Since G' is a subgraph of G_2, $G_2 + {e'}_k$ is also cyclic. So, e_k is an edge of G_2. It contradicts that e_k is not in G_2. Therefore each e_i is an edge of G_2. Similarly, each f_i is an edge of G_1. So, $G_1 = G_2$ and G_n^* is well defined. □

In Fig. 11(a), there exist two alternative constraints $(b_3, l_3) \oplus (l_3, b_2)$ and $(b_4, l_3) \oplus (l_3, b_2)$. Since $G'_n + (l_3, b_2)$ is cyclic, $G_n^* = G'_n + (b_3, l_3) + (b_4, l_3)$.

G_n^* is also a subgraph of G_n if the netlist is monotonic. We consider following lemma to show it.

Lemma 1: Let a_k and a'_k be edges of alternative constraints $a_k \oplus a'_k$ in monotonic orthogonal netlist, and let G be a subgraph of G_n. a_k is in G_n if $G + a'_k$ is cyclic.

Proof. If the orthogonal netlist is monotonic, then monotonic routing patterns exist. Let G_R be the routing pattern graph for a monotonic routing pattern. Let G be a subgraph of G_n such that $G + a'_k$ is cyclic. Since G is also a subgraph of G_R, $G_R + a'_k$ is cyclic. Therefore, G_R does not have a'_k since G_R is acyclic. G_R has either a_k or a'_k, hence G_R has a_k. Since the routing pattern graph for any monotonic routing pattern has a_k, G_n has a_k. □

Theorem 6: An orthogonal netlist is not monotonic if G^*_n is cyclic.

Proof. We show that if an orthogonal netlist is monotonic, then G^*_n is acyclic. If the orthogonal netlist is monotonic, then G^*_n is a subgraph of G_n by Lemma 1 since G'_n is a subgraph of G_n and edges added are edges in G_n. Therefore, G^*_n is acyclic since G_n is acyclic. □

An alternative constraint $a_i \oplus a'_i$ is said to be undecided if $G^*_n + f$ is acyclic for any $f \in \{a_i, a'_i\}$. Assume that **N** is a monotonic orthogonal netlist, and there are c' undecided alternative constraints. Let $a_i \oplus a'_i$ $(1 \le i \le c')$ be undecided alternative constraints. Let G_R be the routing pattern graph for a monotonic routing pattern of **N**. Since either a_i or a'_i is in G_R, there exists a combination $\{f_1, f_2, \ldots, f_{c'}\}$ of alternative constraints such that $G^*_n + \{f_1, f_2, \ldots, f_{c'}\}$ is acyclic, where $f_i \in \{a_i, a'_i\}$. Therefore an orthogonal netlist is not monotonic if $G^*_n + \{f_1, f_2, \ldots, f_k\}$ is cyclic for any combination $\{f_1, f_2, \ldots, f_k\}$ of undecided alternative constrains. This constraints should be analyzed thoroughly in our future work since the number of combinations is exponential in terms of the number of undecided alternative constraints.

4.2 An Orthogonal Routing Method

In this section, we present a monotonic orthogonal routing method. Similar to parallel routing method, we obtain an order from an order graph, and each route is generated one by one according to the order. But, it has several different points. Firstly, we need to consider alternative constraints and undecided alternative constraints should be decided without generating a cycle in the order graph. Though our algorithm try not to generate cycles, an order graph may become cyclic depending on a combination of undecided alternative constraints since each undecided alternative constraint is decided one by one. Secondly, it is not guaranteed that a monotonic routing pattern can be obtained even if we can construct an acyclic order graph with edges of all alternative constraints, and generate routes according to an order obtained from the order graph. Non-monotonic route might be generated due to the unknown constraints between four more nets, as if order b_2, l_3, and b_3 for the netlist in Fig. 12(b) was obtained without considering alternative constraints. In this paper, we give a simple method for orthogonal netlist, though we should investigate whether monotonic routing pattern can be obtained or not if an acyclic order graph is obtained which is in our future work. More practical algorithm will be obtained if more detailed analysis of alternative constraints is given and the density is considered.

In our order selection method first, order graph G'_n is constructed by necessary conditions without alternative constraints. Then, edges corresponding to alternative constraints is added if the decision is possible. This procedure is shown in Fig. 13 as Alternative Decision. The graph obtained by alternative decision is a subgraph of G^*_n since some edges which can be added to G^*_n may not be added.

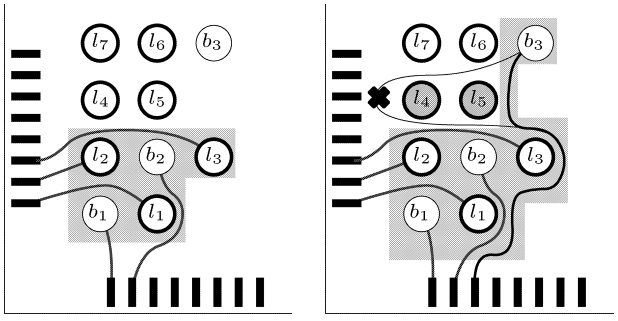
In alternative decision, whether an order graph becomes cyclic by adding either edge or not is checked for each alternative constraint at most once. Cyclic check needs $O(n + m)$. So, the time complexity in alternative decision is $O(n^3 + n^2 m)$ since the number of alternative constraints is $O(n^2)$. Since alternative decision is applied after each source removal, the time complexity finding order is $O(n^4 + n^3 m)$. But, in Sect. 5 we show that our method can obtain a monotonic routing pattern speedy.

According to the obtained order, fingers are connected

```
Order_Selection(Netlist N) {
    create order graph G'_n from N
    G ← G'_n
    A ← the set of alternative constraints for N
    G ← Alternative_Decision(G, A)
    S ← a set of the sources in G
    while S ≠ ∅ {
      select v ∈ S
      remove the vertex v and edges connected to v in G
      for each (u1, u2) ⊕ (u3, v) ∈ A
        G ← G + (u1, u2)
      if alternative constraints corresponding to v exist,
                                    then they are removed from A
      G ← Alternative_Decision(G, A)
      S ← a set of the sources in updated G
    }
    if G is empty, return the obtained order of nets
    otherwise return failure status
}

Alternative_Decision(Graph G, alternative constraints A) {
  for each a_i ⊕ a'_i
    if G + a_i is cyclic,
                    then G ← G + a'_i, and remove a_i ⊕ a'_i from A
  return G
}
```

Fig. 13 Order selection algorithm.



(a) Before b_3 is realized. (b) After b_3 is realized.

Fig. 14 An example of making routes.

Table 1 The results of experiments. (5000 patterns in each size)

#(Net)	64	121	256	529	1024	2025
#(Failure)	0	0	1	1	6	7
Success Rate [%]	100	100	99.98	99.98	99.88	99.86
#(Alt. Min.)	0	0	5	15	25	66
#(Alt. Ave.)	7.1	14.1	28.6	53.4	94.6	171.0
#(Alt. Max.)	37	71	125	277	336	651
Time [sec]	0.003	0.007	0.021	0.059	0.175	0.609

to balls by monotonic routes one by one from the lower left. Formally, routing of a bottom net is defined as follows: A ball is said to be *connected* if its route is completed. Otherwise, a ball is said to be *unconnected*. Let b be a bottom net. $R(b)$ passes as the left as possible on condition that $R(b)$ passes to the right of the unconnected left net balls in the lower-left region of b and connected balls. For example, consider $R(b_3)$ in Fig. 14. $R(l_4)$ and $R(l_5)$ become non-monotonic if $R(b_3)$ passes to the left of them. Therefore, $R(b_3)$ needs to avoid balls of nets l_4 and l_5 as shown in Fig. 14(b). Similarly, routes of left nets can be decided.

5. Experiments and Results

We implemented our method for orthogonal netlists with C++ language and applied it to monotonic orthogonal netlists in order to investigate success rate since it is not guarantee that our method for orthogonal netlist completes routing. The program ran on a personal computer with a 3.4 GHz CPU and 1 GB of memory.

Monotonic orthogonal netlists are generated by relaxing the sufficient condition in Sect. 4.1.1. We applied our method to problems of 6 sizes from 8×8 to 45×45. 5000 patterns were generated in each size. Results are shown in Table 1. In Table 1, the number of netlists that monotonic routing patterns are not obtained and the ratio that monotonic routing patterns are obtained for each size are shown in second and third row, respectively. The minimum, average, and maximum number of alternative constraints and average computation time per netlist for each size are shown in from fourth row to seventh row, respectively. An example of output is shown in Fig. 15.

In this experiment, all of failures occur because the order graph becomes cyclic due to a bad selection of alternative constraints. A monotonic routing pattern may not be obtained even if an order is obtained by our algorithm shown in Fig. 13. However, a monotonic routing pattern is obtained when an order is obtained in this experiment.

The number of alternative constraints increases linearly in teams of the number of nets in this experiment. However, the success rate is more than 99% and a monotonic routing pattern is obtained within 1 second even if the number of nets is more than 2000. For current problem size, the execution time and success rate are enough.

The practical algorithm will be obtained if the density is taken into account in order selecting.

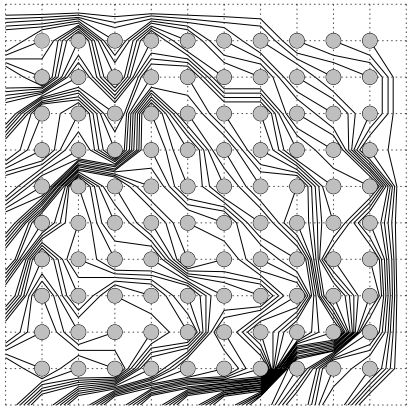


Fig. 15 An example of output. (100 nets)

6. Conclusion

We gave the necessary and sufficient condition for parallel netlist being monotonic, and proposed a routing method for monotonic parallel netlists based on this condition. Moreover we gave a necessary condition and a sufficient condition for orthogonal netlists being monotonic, and proposed a routing method for monotonic orthogonal netlists based on the necessary condition.

As our future work, we need to investigate alternative constraints and whether constraints between four or more nets exist or not. Routing methods that take routing density into consideration should be proposed. Moreover, the method taking non-monotonic routes into account should be proposed. In the method, the monotonic routes obtained by our proposed method is used as an initial solution and is improved iteratively to satisfy the design rule.

Acknowledgments

The authors are indebted to Dr. Kubo and Mr. Kohira for their valuable advices. Their advices are very helpful to us in proposing monotonic parallel and orthogonal routing methods with order graphs. This research was partially supported by Japan Society for the Promotion of Science (JSPS), Grant-in-Aid for Scientific Research (C) 18500034, 2006.

References

[1] E.S. Kuh, T. Kashiwabara, and T. Fujisawa, "On optimum single-row routing," IEEE Trans. Circuits Syst., vol.CAS-26, no.6, pp.361–368,

1979.
[2] S. Tsukiyama and E.S. Kuh, "Double-row planar routing and permutation layout," Networks, vol.12, no.3, pp.287–316, 1982.
[3] M.F. Yu and W.W.M. Dai, "Single-layer fanout routing and routability analysis for ball grid arrays," Proc. International Conference Computer-Aided Design, pp.581–586, 1995.
[4] S. Shibata, K. Ukai, N. Togawa, M. Sato, and T. Ohtsuki, "A BGA package routing algorithm on sketch layout system," J. Jpn. Inst. Interconnect. Packag. Electron. Circuits, vol.12, no.4, pp.241–246, 1997.
[5] Y. Kubo and A. Takahashi, "A global routing method for 2-layer ball grid array packages," Proc. ACM International Symposium on Physical Design, pp.36–43, April 2005.
[6] 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 Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.17, no.2, pp.182–189, 1998.
[7] S.S. Chen, J.J. Chen, C.C. Tsai, and S.J. Chen, "An even wiring approach to the ball grid array package routing," Proc. International Conference on Computer Design, pp.303–306, 1999.

Yoichi Tomioka received his B.E. degree in computer science, from Tokyo Institute of Technology, Tokyo, Japan, in 2005. He is currently a master course student of the Department of Communications and Integrated Systems in Tokyo Institute of Technology. He will receive his M.E. degree in 2006, and will be a doctor course student of the department. His research interests are in VLSI package design automation and combinational algorithms.

Atsushi Takahashi received his B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He had been with the Tokyo Institute of Technology as a research associate from 1991 to 1997 and has been an associate professor since 1997. He visited University of California, Los Angeles, U.S.A., as a visiting scholar from 2001 to 2002. He is currently with the Department of Communications and Integrated Systems, Graduate School of Science and Engineering, Tokyo Institute of Technology. His research interests are in VLSI layout design and combinational algorithms. He is a member of IEEE and IPSJ.