Simultaneous Escape Routing on Multiple Components for Dense PCBs

Ching-Yu Chin and Hung-Ming Chen
Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan

***Abstract*—Simultaneous escape routing on dense boards is a very challenging task and lots of efforts have been spent. Most previous works focus on routing with one or two components, but in reality more components are possibly involved. In this paper, we propose a new simultaneous escape routing algorithm that considers multiple components at once. The proposed routing algorithm can solve simultaneous escape problem when two or more busses are connected to the same component. The experimental results show that, our routing algorithm can efficiently arrange escape paths properly to avoid net crossings and blockings between different busses.**

I. Introduction

Today we have increasing pin count on the components, thus we have the very dense boards. Escape routing and PCB area routing has become important issues in off-chip design. Escape routing problem is to route pins inside a component, such as ball grid array (BGA), to the boundary of the component. It is then followed by PCB routing, which routes the escaped nets on board. Usually, PCB routing is routed by bus, which is assumed that nets within a bus are escaped in a proper order. However, due to the rapidly increasing density of components, it is a challenge on arranging escape paths. Since the solutions of escape routing directly affect PCB area routing, many research efforts have been put on the escape routing.

There are several types of escape routing problem. *Single component escape routing* considers only one component at a time. This type of routing can be characterized as *ordered escape routing* and *unordered escape routing*. Ordered escape routing restricts the escape order of the nets. To meet the required escape order, ordered escape routing algorithm may produce longer wirlength, or consume more routing area. [8] [9] solve the ordered escape problem by the formulation of SAT problem. On the other hand, unordered escape routing usually utilizes flow based approach to maximize the number of routable nets, [2] and [3] are on single escape routing.

Simultaneous escape routing takes two sides (two components) of a bus into consideration simultaneously. In this type of routing, the escape order is not limited, but is negotiated by the pin locations of the two components. The advantage of simultaneous escape routing is that the escape order is not restricted, they can use routing resource more efficiently than ordered escape routing. In addition, for the target bus, it requires a common order between the two components, therefore it also solves the potentially occurred ordered problem in single component escape. [1] and [7] focus on this type of problem. Finally, *free assignment escape routing* routes two sides of a bus simultaneously, but the pins are not pre-assigned. That is, a pin can belong to any unrouted net. The free assignment escape routing is similar to single escape routing, they have more flexibility on arranging routing path than ordered/simultaneous escape routing. [4] tackles with this type of routing problem.

Generally, a component is possible to connect to more than one components, which means there are two or more busses needed to be escaped to the component boundary. Because the routing resource is limited inside the component, it is difficult to arrange escape paths of different busses blocking each other. Most works mentioned above focus on routing problem under one or two components, which cannot take this problem into consideration. [5] proposes an algorithm to find the maximum disjoint subset of rectangles. It utilizes the technique to figure out which busses in a component can be escaped in the same layer. However, the algorithm restricts routing area of a bus to rectangle, and rectangle of bus cannot overlap with other busses. This strategy uses the routing resource inefficiently, especially inside the limit area of a component.

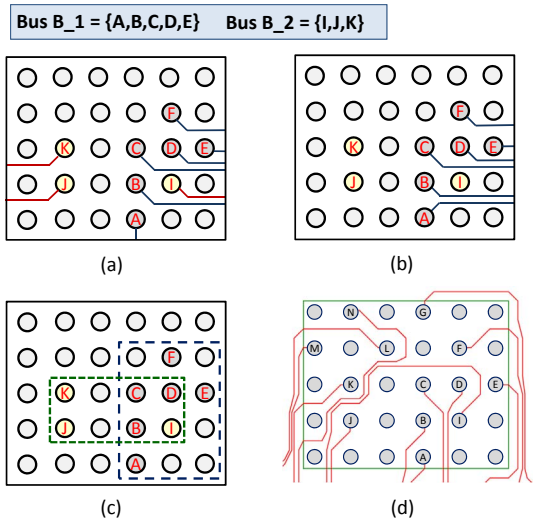

Fig. 1. An illustration of routing problem when multiple busses of a component need to be escaped. (a) The routing result is illegal: net I belongs to B_2 but not B_1, (b) The routing result of B_1, net I is blocked, (c) The rectangle representation of the two busses, they are overlapped and cannot route in the same layer in [5]. (d) The routing result of the proposed algorithm.

Fig. 1 illustrates the situation of not taking adjacent busses into consideration. Two busses $B_1 = \{ABCDE\}$ and $B_2 = \{IJK\}$ are in the same component. Because B_1 and B_2 are different busses (connect to different components), the escape orders of the nets of the component cannot be interleaving, i.e. the routing result shown in Fig. 1(a) is not a legal solution. Furthermore, if we route the two busses separately, it may result in net blocking as shown in Fig. 1(b). Net I is blocked by paths of B_1 since we do not consider two busses simultaneously. Fig. 1(c) shows the rectangle representation of bus used in [5], it is because the rectangles of the two busses are overlapped, the situation will be treated as unroutable in single layer. However, in many cases, pin locations are clustered by bus with few pins locate separately. In fact, it is able to route several busses in one layer. Fig. 1(d) shows the routing result of B_1 and B_2 from our proposed algorithm. We can see that pin I is detoured with all nets are routed successfully.

In this paper, we propose a heuristic algorithm to solve the escape routing problem of three or more components. Our routing algorithm consists two main stages:

1) Escape order generation, which generates a set of feasible escape order candidates.
2) Escape path finding stage, which routes the to-be-escape pins to component boundary according to the order

978-1-4799-2311-3/13/$31.00 ©2013 IEEE

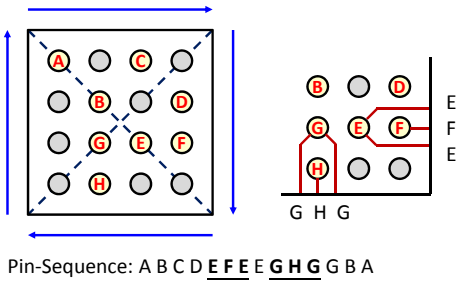

Fig. 2. The pin-sequence of a component. The component is divided into four areas: top, right, bottom, and left by the dotted lines. Pin which locates on the dotted lines belongs to areas of both sides of the line.

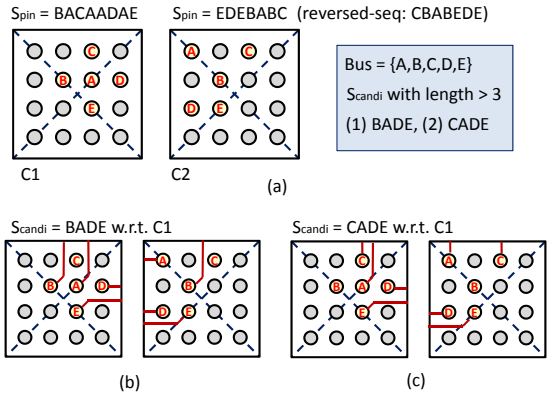

Fig. 3. The candidate-sequence of a bus. (a) Pin locations and corresponding P_{seq} of the two components. (b) Routing result of S_{candi} = $BADE$ w.r.t. $C1$. (c) Routing result of S_{candi} = $CADE$ w.r.t. $C2$.

generated in stage one.

The algorithm takes routing of different busses into consideration simultaneously, and generates feasible routing results in an acceptable execution time. The rest of this paper is organized as follows: Section II gives the description of the routing algorithm. Section III shows the experimental results of the proposed routing method. Section IV concludes the paper.

II. Problem and Proposed Algorithm

The input of the multiple component escape routing problem is the placement of components, netlist, and pin locations. Each component is a ball grid array, with some balls are specified as to-be-escape pins.

Our routing algorithm consists two stages: (1) escape order generation and (2) path finding. In the order generation stage, we generate a set of feasible "partial" order candidates. The term "partial" means that the order generated in this stage is incomplete. For example, an order candidate of bus $\{ABCDE\}$ might be BAC with net D and E are not specified in the order. Pins involved in the order candidates indicate that, according to the order, they can be routed in monotonic paths without any crossing. In the path finding stage, we employ and modify the dynamic routing graph in [11] as our routing graph. The detailed explaination of the proposed algorithm are described in the following subsections.

A. Escape Order Generation

In this subsection, we introduce our escape order generation strategy. For a given BGA placement, netlist and pin locations, we define the following parameters: pin-sequence, candidate-sequence, and order-candidate.

Pin-Sequence: A pin-sequence (S_{pin}) of a component is an ordered sequence generated by the manner illustrated in Fig. 2. S_{pin} of a component contains all feasible escape orders that routes in monotonic paths.

Take Fig. 2 as an example, the BGA is divided into four areas: top, left, bottom, and right. Each ball belongs to one to four regions. We randomly select a pin of the BGA as the start point of the sequence. In this case, we select pin A as the start point. From this start point, other pins of the BGA are inserted into the sequence in the following manner. For pins located in top area, which are A, B, and C, the corresponding sequence is simply ABC because the three pins are separated in different columns. For pins located in the right area, the corresponding sequence is $DEFE$ since E and F are located in the same row. For pins located in the bottom area, the rule is the same as in top area with reversed direction. G and H are in the same column, therefore the corresponding sequence is $EGHG$. The sequence of the left area is GB. Finally, we have the pin sequence of the target component: $ABCDEFEEGHGGB$.

Candidate-Sequence: A candidate-sequence (S_{candi}) of a bus is an ordered net sequence which indicates a feasible partial escape order of the bus with monotonic paths. That is, for a bus with three nets, the length (or size) of its order-candidate may be equal or less to three, and one bus corresponds to several S_{candi}s. S_{candi} can be generated from the two pin-sequences the target bus connects to.In order to represent S_{candi} of a bus, the ordered sequence is counted clockwise with respect to one of the components the bus connects with. Here we define S_{candi} to be represented as the following format (which means nets of the sequence is listed clockwise ($Sequence$): with respect to component COMP.)
$\{Sequence\}$ w.r.t. COMP.

Order-Candidate: An order-candidate (O_{candi}) of a set of busses is a feasible partial escape order of these busses, in which there exists no crossings between any pair of nets.

The pin-sequence of a component contains the combinations of monotonic routing paths of the component. For instance, the first symbol E in $S_{pin} = \{ABCD\underline{E}FEEGHGGB\}$ of Fig. 2 represents a routing path of E, which escapes toward right above pin F. And the first G of the S_{pin} represents a routing path toward down at the right hand side of H.

Because each two-terminal bus connects to two components, the escape order of the bus is affected by the two corresponding pin-sequences. Assume bus B connects to component C_1 and C_2 with S_{pin} S_{p1} and S_{p2}. A candidate-sequence of B must be a common subsequence of S_{p1} and reverse S_{p2} in a cyclic manner, because S_{pin} involves all possible escape order of pins with monotonic routing paths. Since S_{candi} represents a partial escape order of a bus, and O_{candi} represents a partial escape order of a set of busses, it is easy to figure out that an O_{candi} of K busses is a set of K S_{candi}, say S_{c1} to S_{cK}, where S_{ci} is a candidate-sequence of bus i. In addition, the monotonic escape paths of these K busses must have no crossing with each other.

Here we use an example to illustrate the generation of O_{candi} of a bus. Fig. 4 shows a design with three components and three busses. First, for each BGA component, we generate its S_{pin}. Second, we split each pin-sequence into segments by bus. As illustrated in Fig. 4, the results of split S_{pin} of C_1 are two different sequences, with each consisting three segments. We apply the same split procedure on component C_2 and C_3, each of them produces one sequence respectively. Without loss of generality, we select the first split sequence of C_1, $\{CDFFD - 12213 - BBA\}$, as a start. According to this sequence, the S_{candi} of bus $B1$ w.r.t to component C_1 with length larger than 2 are $\{BACD, BAC, BCD, BAD, ACD\}$. S_{candi} of bus $B2$ w.r.t. to component C_2 (with length larger

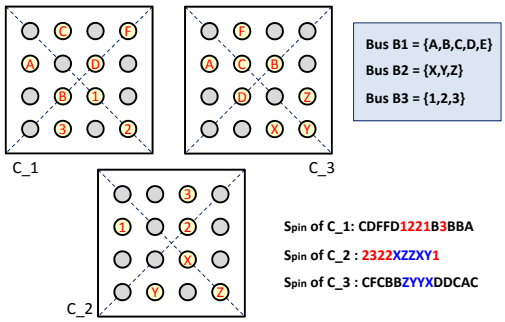

Fig. 4. The sample design and their pin-sequences.

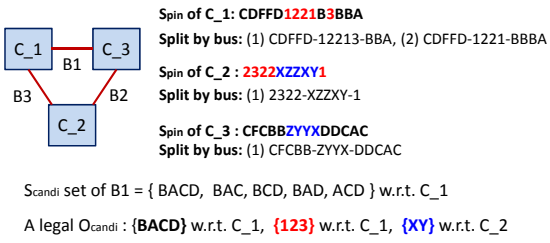

Fig. 5. An example of O_{candi} generation of a three component escape problem in Fig. 4. The legal solution shows that we can easily route nets that are involved in the candidate without any detour.

than 1) are $\{XY, ZY\}$. And S_{candi} of bus $B3$ w.r.t. to component C_1 (with length larger than 2) is $\{123\}$. From the above descriptions, we can obtain one order-candidate $\{BACD, 123, XY\}$, the length is equal to 9. We can also obtain other O_{candi} with different length, or from different split sequences.

Note that an order-candidate is a possible escape order of given busses, we can easily route nets that are involved in the candidate without any detour.

B. Routing On Dynamic Routing Graph

In this paper, we adopt and modify the dynamic routing graph in [11] to generate our escape routing graph. In [11], a dynamic routing graph is proposed to solve the PCB bus planner problem. The graph is "dynamic" because it updates nodes and edges whenever a new path is routed. The advantage of the graph is that it produces planar routing topology without calculating detailed path locations. To adopt dynamic graph on escape routing, we have to define (1) the initial routing graph (before any node/edge updating), and (2) the update rules of the graph.

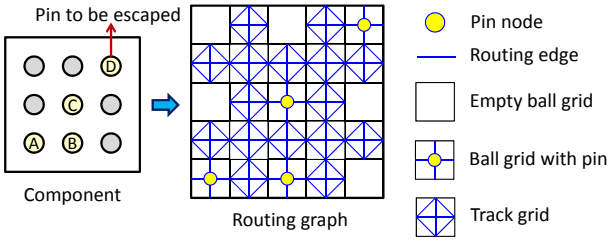

Fig. 6. The routing graph of a component with $3*3$ ball array and 4 escape pins.

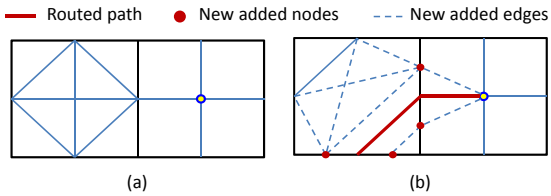

Fig. 7. (a) The initial routing graph of a pin grid and a track grid. (b) The updated routing graph after a path passes through the two grids.

We construct the routing graph $G_i(V, E)$ of a BGA C_i as illustrated in Fig. 6. A BGA with M*N ball array is divided into $(2M-1)*(2N-1)$ grids, each grid is either a ball grid or a track grid. For a ball grid, it has a node belonging to V in the center if the ball is a pin, otherwise it is an empty grid. The node is called the *pin node*. For a track grid, it has four nodes that are on the middle point of the four sides of the grid boundary. In track grid, edges are connected to nodes in same grid; while in ball grid, edges are connected between the pin node and nodes on grid boundaries. Fig. 6 shows the initial routing graph of a component with 3×3 ball array.

When routing on the dynamic graph, once a path is routed, the graph will update itself by adding extra nodes and edges. Fig. 7(a) shows the initial routing graph in two grids (a pin grid and a track grid), and Fig. 7(b) shows the updated routing graph of the grids after a path p has passed through. The routed nodes and edges can no longer be used, and edges cross with the routed path are prohibited. For a design with K components, a *complete routing graph* is generated by the union of the K routing graphs G_1 to G_K. Extra nodes and edges are added to connect these graphs. The path finding procedure is performed on the complete graph.

In Section II-A, we obtain O_{candi}. Nets involved in O_{candi} can be routed in monotonic path directly on the constructed dynamic graph of single component. The resultant path is from pins to component boundaries, which are the escape paths of these pins. Next, we route each of these escaped nets from component boundary to another boundary, that is, to connect each net into a single path. This step ensure the correctness of the later routing step: the routing of nets not belonging to O_{candi}. Also, this step illustrates us the possible topology of PCB area routing. For nets not belonging to the O_{candi}, we apply A* algorithm to route these nets on the complete dynamic graph. In our implementation, the capacity between two ball grids is two.

The dynamic routing graph finds path without giving a real path location, but generates a topology of the path. Consequently, the resulting routing solution is planar. And routing paths inside component is the escape paths of the nets.

C. Overall Flow of Our Approach

Fig. 8 shows the overall flow of the proposed algorithm. Given Component placements, netlist, and pin locations, we first generate the ordered-candidate set using pin-sequences. All candidates are sorted in decreasing order of their length (the number of nets). Second, we construct the dynamic routing graph of each components. For each candidate, nets belonging to the candidate are routed first. If there exists any unrouted net, we will route the remaining nets on the complete graph. The procedure continues until there is no unrouted nets left.

III. Experimental Results

Our algorithm was implemented in C++ programming language on a Intel(R) Xeon(R) Linux workstation with 16GB memory. We have constructed the test cases which are based

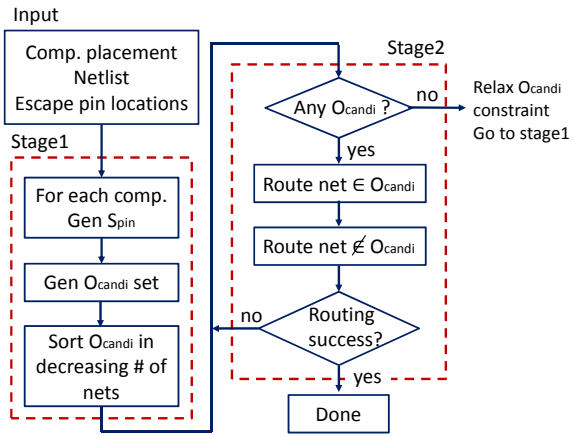

Fig. 8. Flow chart of the proposed escape routing algorithm.

TABLE I
THE SUMMARY OF TEST CASES AND EXPERIMENTAL RESULTS.

	#Nets	#Comp.	#Busses	Total Time
Test Case I	15	2	1	<1 sec
Test Case II	20	3	3	2 sec
Test Case III	31	3	3	2 sec
Test Case IV	26	4	4	2 sec
Test Case V	34	4	4	3 sec

on cases shown in recent industrial works/references. The capacity between two adjacent balls are two.

Table. I shows the test cases and routing efficiency for our router. $Nets$, $Comp.$, and $Busses$ indicate the number of nets, components, and busses in test cases, respectively. $Total\ Time$ includes the time for building the routing grid and the run time spent by our router.

Fig. 9 shows the routing result of Test Case II. There are three components and 20 nets in this case. Those 20 nets belong to three different busses, $B1 = \{ABCDEFG\}$, $B2 = \{IJKLMN\}$, and $B3 = \{PQRSTUV\}$. The router routes these three busses simultaneously and produces feasible solutions under capacity constraint within seconds. Note that

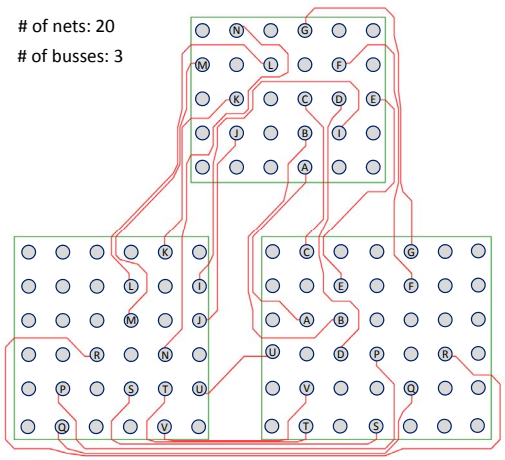

Fig. 9. Routing result of Test Case II

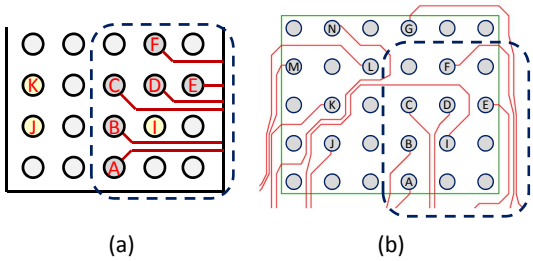

Fig. 10. (a) A routing result of bus $\{ABCDE\}$ that causes net I unroutable. (b) An enlarged view of path arrangement produced by the proposed routing algorithm of net I and nets surrounding to it.

pin I belongs to $B2$, but its location is surrounding with pins of $B1$. The router prevents the net path from blocking by those nets. Fig. 10 shows the enlarged view of the escape path of I in Test Case II. The left hand side of the figure illustrates a routing result of bus $\{ABCDE\}$ that causes net I unroutable. For conventional escape routing, it may produce routing results similar in Fig. 1(b). Our proposed routing algorithm avoids the situation and generates feasible solution.

The O_{candi} of the routing result of Test Case II contains 17 nets. Since the remaining three nets D, I, and N do not show up in the corresponding O_{candi}, these nets are routed freely on dynamic graph without constraints on their escape direction, and therefore are not trapped due to their disadvantageous locations. During the routing procudure, some O_{candi} produce inevitable net blockings, these infeasible results are abandoned. Our final result shows 100% routability.

IV. Conclusion

In this paper, we propose a heuristic algorithm to tackle the simultaneous escape routing problem on multiple components. The algorithm uses partial order candidates to explore possible solutions, and adopts dynamic routing graph to route escape nets. The proposed approach can take multiple busses and components into considerations, avoid possible crossings or blockings between nets of different busses.

References

[1] M. M. Ozdal and D. F. Wong, "Simultaneous Escape Routing and Layer Assignment for Dense PCBs," In IEEE/ACM International Conference on Computer-Aided Design, pp.822-829, 2004.
[2] T. Yan and D. F. Wong, "A Correct Network-Flow Model for Escape Routing," In Design Automation Conference, pp.332-335, 2009.
[3] T. Yan, P.-C. Wu, Q. Ma, and D. F. Wong, "On the Escape Routing of Differential Pairs," In IEEE/ACM International Conference on Computer-Aided Design, pp.614-620, 2010.
[4] H. Kong, T. Yan, and D. F. Wong, "Optimal Simultaneous Pin Assignment and Escape Routing for Dense PCBs," In Asia South-Pacific Design Automation Conference, pp.275-280, 2010.
[5] H. Kong, Q. Ma, T. Yan, and D. F. Wong, "An Optimal Algorithm for Finding Disjoint Rectangles and Its Application to PCB Routing," In Design Automation Conference, pp.212-217, 2010.
[6] Q. Ma, T. Yan, and D. F. Wong, "A negotiated Congestion Based Router for Simultaneous Escape Routing," In International Symposium on Quality Electronic Design, pp.606-610, 2010.
[7] L. Luo, T. Yan, Q. Ma, and D. F. Wong, "B-escape: A Simultaneous Escape Routing Algorithm Based on Boundary Routing," In International Symposium on Physical Design, pp.19-25, 2010.
[8] L. Luo and D. F. wong, "Ordered Escape Routing Based on Boolean Satisfiability," In Asia South-Pacific Design Automation Conference, pp.244-249, 2008.
[9] L. Luo and D. F. Wong, "On Using SAT to Ordered Escape Problems," In Asia South-Pacific Design Automation Conference, pp.594-599, 2009.
[10] Y. Tomioka and A. Takahashi, "Monotonic Parallel and Orthogonal Routing for Single-Layer Ball Grid Array Packages," In Asia South-Pacific Design Automation Conference, pages 642-647, 2006.
[11] H. Kong, T. Yan, and D. F. Wong, "Automatic Bus Planner for Dense PCBs," In Design Automation Conference, pp. 326-331, 2009.
[12] H. Kong, Q. Ma, T. Yan, and D. F. Wong, "An Optiamal Algorithm for Finding Disjoint Rectangles and Its Application to PCB Routing," In Design Automation Conference, pp.212-217, 2010.
[13] J.-W. Fang, C. Hsu, and Y.-W. Chang, "An Integer Lnear Programming Based Routing Algorithm for Flip-Chip Design," In Design Automation Conference, pages 606-611, 2007.
[14] J. W. Fang, K.-H Ho, and Y.-W. Chang, "Routing for Chip-Package-Board Co-Design Considering Differential Pairs," In Proc. of IEEE/ACM International Conference on Computer-Aided Design, pp.512-517, 2008.