Simultaneous Escape Routing Based on Routability-Driven Net Ordering

Jin-Tai Yan, Tung-Yen Sung and Zhi-Wei Chen
Department of Computer Science and Information Engineering, Chung-Hua University
Hsinchu, Taiwan, R.O.C

Abstract

In this paper, given a set of *n* escape nets between an array of *p*x*q* pins and an array of *r*x*s* pins, firstly, a routability-driven net order between two given pin arrays is determined for simultaneous escape routing. Furthermore, based on ordered escape routing for two pin arrays, an efficient approach is proposed to solve the routing problem for simultaneous escape routing. Compared with Kong's flow-based approach[11] for three tested examples, the experimental results show that our proposed approach achieves 100% routability for the tested examples and reduces the CPU time by 54.1% on the average.

I. INTRODUCTION

Due to high pin counts in modern ICs, board-level routing becomes more and more difficult in modern PCB designs. For board-level routing, escape routing is a key problem and it becomes important to develop an efficient routing approach. For escape routing, the problem can be divided into *unordered escape routing* and *ordered escape routing*. In unordered escape routing, there exists no constraint on the order of the escape pins. Many research results[1-3] are contributed on the problem. It is known that the unordered problem can be optimally solved by using a network-flow-based approach. In ordered escape routing, the boundary pins can be further divided into fixed pins and floating pins according to the location feature of the boundary pins. In ordered escape routing with fixed pins, all the boundary pins are located on fixed locations. It is known that all existing approaches[4-5] cannot guarantee to find a routing solution even if one exists. Besides that, the proposed approach in [4] only routes nets in a monotonic detour-free style and the recent approach in [5] considers the non-monotonic routing only when the routing graph has a cycle. Even though there exists no cycle in a routing graph, the approach[5] may not find the routing solution where detour is needed to release the resource conflict. On the other hand, in ordered escape routing with floating pins, all the boundary pins are not located on fixed locations. Recently, two SAT-based approaches[6-7] are proposed to exactly complete ordered escape routing based on Boolean satisfiability. Although the SAT-based approaches perform well in routability, the proposed approaches do not consider the cases with the larger capacity between any pair of adjacent pins. Besides that, the SAT-based approaches take more CPU time to solve the problem. To solve the routing cases with larger capacity, a heuristic approach[8] based on hierarchical bubble sorting is proposed to assign routability-driven pins for ordered escape routing. Furthermore, ordered escape routing can be completed by using single-layer global routing and grid-based detailed routing.

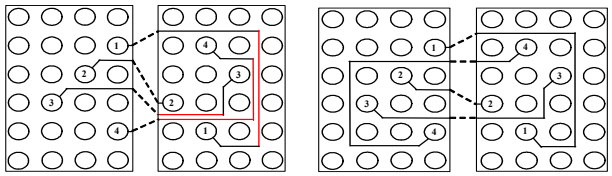
For simultaneous escape routing between two adjacent pin arrays, a graph-based approach[9] is firstly proposed to find the maximal escape nets in a single layer. However, the routing result depends on the span of escape routes in escape nets. Besides that, the flow-based approach[11] is further proposed to escape all the pins of two components independently by using *unordered escape routing* in a correct network flow model[10] and then assign pins to nets by sweeping escape positions along the boundaries of two components in reverse orders. However, the proposed approach has higher time complexity to solve the problem. Recently, a negotiated congestion-based routing scheme[12] is used to build a routing graph for simultaneous escape routing. Although the proposed approach obtains better solutions for some specific examples, it is more time-consuming for general examples. Besides that, based on boundary routing, a new heuristic approach[13] is proposed for simultaneous escape routing. It is clear that the proposed approach is based on dynamic net ordering to solve the problem. However, it uses more time to find a routable net order.

In general, a straightforward solution can be obtained by running *unordered escape routing* for one component and running *ordered escape routing* for the other component. However, a worse net order between two adjacent pin arrays may yield the unroutable result in simultaneous escape routing. Hence, it is important for simultaneous escape routing to find a routability-driven net order between two adjacent pin arrays. Figure 1 gives such an example whose 4 escape nets are expected to be connected in two pin arrays with the capacity 1 for simultaneous escape routing. In Fig. 1(a), 4 nets are escaped in the left pin array and the net order is obtained as 1->2->3->4. Furthermore, the net order is used in the right pin array for ordered escape routing. However, the ordered escape routing cannot find a feasible solution with the capacity 1 for the right pin array. In Fig. 1(b), if a routability-driven net order, 1->4->2->3, is used for two pin arrays, a feasible solution with the capacity 1 can be obtained in simultaneous escape routing.

In this paper, given a set of *n* escape nets between an array of *p*x*q* pins and an array of *r*x*s* pins, firstly, a

978-1-4577-1617-1/11/$26.00 ©2011 IEEE

routability-driven net order between two given pin arrays is determined for simultaneous escape routing. Furthermore, based on ordered escape routing for two pin arrays, an efficient approach is proposed to solve the routing problem for simultaneous escape routing. Compared with Kong's flow-based approach[11] for three tested examples, the experimental results show that our proposed approach achieves 100% routability for the tested examples and reduces the CPU time by 54.1% on the average.



(a) (b)

Fig. 1 Routability-driven net order in simultaneous escape routing

II. PROBLEM FORMULATION

For simultaneous escape routing, it is assumed that $A1=\{P_{1,1}, P_{1,2},..., P_{1,q},..., P_{p,1}, P_{p,2},..., P_{p,q}\}$ is a given array of pxq pins with capacity, c_1, under a component, $A2=\{P_{1,1}, P_{1,2},..., P_{1,s},..., P_{r,1}, P_{r,2},..., P_{r,s}\}$ is a given array of rxs pins with capacity, c_2, under the other component and $N=\{N_1, N_2,..., N_n\}$ is a set of n escape nets in a single-layer bus between $A1$ and $A2$. It is known that any escape net only has one escape terminal inside $A1$ and the other escape terminal inside $A2$. In general, simultaneous escape routing is only allowed to route all the escape nets between $A1$ and $A2$ in a single layer. Hence, a *non-crossing* constraint between any pair of escape nets must be maintained for simultaneous escape routing. On the other hand, a *capacity* constraint between any pair of two adjacent pins must be maintained because the space between any pair of two adjacent pins is limited.

Given a set of n escape nets, N, between two pin arrays, $A1$ and $A2$, under two adjacent components and two capacities, c_1 and c_2, in two pin arrays, $A1$ and $A2$, the simultaneous escape routing(SER) problem is to find n disjoint paths to route the given n escape nets in a single-layer bus between two pin arrays with satisfying the given non-crossing and capacity constraints. In general, any connecting side between two given components is called as an escape side for the SER problem. As illustrated in Fig. 2, two given arrays of 9x10 pins with 14 escape nets are considered for simultaneous escape routing. It is assumed that the capacity between two adjacent pins in two pin arrays is 2. If the required order of the 14 escape nets between two pin arrays is 14->6->2->5->3->10->11->4->1->9->8->7->12->13, the final routing result of the 14 given escape nets with 100% routability can be obtained to satisfy the non-crossing and capacity constraints in a single layer for simultaneous escape routing.

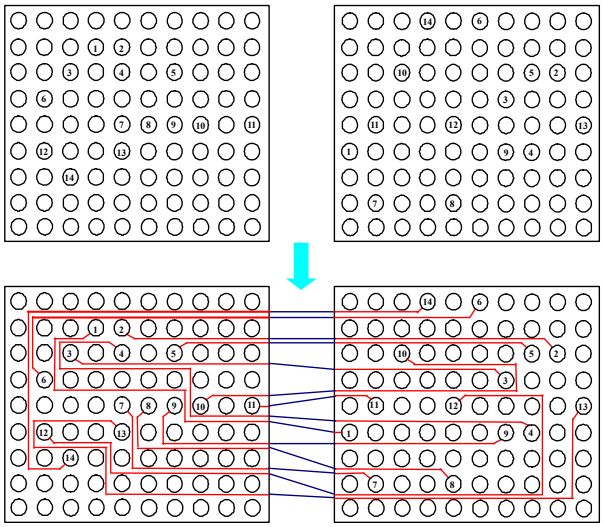


Fig. 2 Simultaneous escape routing for 14 escape nets

III. SIMULTANEOUS ESCAPE ROUTING

Given a set of escape nets in a single-layer bus between two pin arrays, based on the determination of routability-driven net ordering, an efficient approach is proposed for simultaneous escape routing and the routing process is divided into five phases: *Determination of routability-driven net ordering*, *Transformation of net numbers*, *Simultaneous escape global routing*, *Simultaneous escape detailed routing* and *Recovery of net numbers* as shown in Fig. 3.

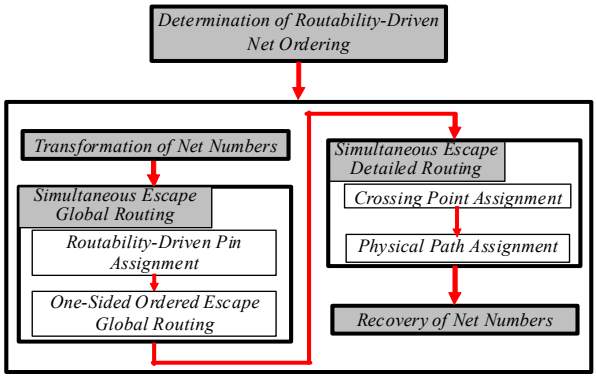


Fig. 3 Design flow for simultaneous escape routing

3.1 Determination of Routability-Driven Net Ordering

In determination of routability-driven ordering, firstly, based on the locations of all the escape terminals in two pin arrays, a connection bipartite graph, $G(V1, V2, E)$, for two pin arrays can be constructed as follows: every vertex in $V1$ represents a set of escape terminals in the same row for the left pin array, every vertex in $V2$ represents a set of escape terminals in the same row for the right pin array and each undirected edge, (u, v), in E, $u \in V1$ and $v \in V2$, represents the connection relation of at least one escape net between two sets of escape terminals. In general, the row-

by-row vertex sets in *V1* and *V2* are sequentially ordered in the bipartite graph, *G(V1, V2, E)*. Refer to the two pin arrays with 14 escape nets in Fig. 2, the corresponding connection bipartite graph, *G(V1, V2, E)* with 6 vertices in *V1*, 6 vertices in *V2* and 12 edges in *E* can be constructed as shown in Fig. 4(a), where $V1=\{V_{\{1,2\}}, V_{\{3,4,5\}}, V_{\{6\}}, V_{\{7,8,9,10,11\}}, V_{\{12,13\}}, V_{\{14\}}\}$, $V2=\{V_{\{6,14\}}, V_{\{2,5,10\}}, V_{\{3\}}, V_{\{11,12,13\}}, V_{\{1,4,9\}}, V_{\{7,8\}}\}$ and $E=\{(V_{\{1,2\}}, V_{\{1,4,9\}}), (V_{\{1,2\}}, V_{\{2,5,10\}}), (V_{\{3,4,5\}}, V_{\{2,5,10\}}), (V_{\{3,4,5\}}, V_{\{3\}}), (V_{\{3,4,5\}}, V_{\{1,4,9\}}), (V_{\{6\}}, V_{\{6,14\}}), (V_{\{7,8,9,10,11\}}, V_{\{2,5,10\}}), (V_{\{7,8,9,10,11\}}, V_{\{11,12,13\}}), (V_{\{7,8,9,10,11\}}, V_{\{1,4,9\}}), (V_{\{7,8,9,10,11\}}, V_{\{7,8\}}), (V_{\{12,13\}}, V_{\{11,12,13\}}), (V_{\{14\}}, V_{\{6,14\}})\}$.

Furthermore, a routability-driven order of all the nets between two pin arrays can be determined by finding a common longest sequence and using a two-sided reassignment process. In finding a common longest sequence, initially, if any vertex in *V1*(*V2*) has at least two connecting edges in the bipartite graph, *G(V1, V2, E)*, the vertex must be separated into some ordered vertices according to the connection relations to the vertices in *V2*(*V1*). As a result, any vertex in the modified bipartite graph, *G'(V'1, V'2, E')*, has only one connecting edge. For two ordered vertex sets, *V'1* and *V'2*, a common longest sequence between *V'1* and *V'2* can be easily found and the nets inside the vertex sequence can not be considered in the two-sided reassignment process. After finding a common longest sequence, if all separated ordered vertices from any vertex in *V1* or *V2* are not selected in the vertex sequence, all the separated ordered vertices must be recovered to form an original vertex and a modified bipartite graph can be constructed. As illustrated in Fig. 4(a), the vertices, $V_{\{1,2\}}$, $V_{\{3,4,5\}}$, $V_{\{7,8,9,10,11\}}$ and $V_{\{12,13\}}$, in *V1* and the vertices, $V_{\{6,14\}}$, $V_{\{2,5,10\}}$, $V_{\{11,12,13\}}$ and $V_{\{1,4,9\}}$, in *V2*, are separated into some ordered vertices. As a result, the nets, 2, 5, 3, 11, 9, 7 and 8, are selected in a common longest sequence.

In two-sided reassignment process, initially, one edge with the most crossings in *G* is selected. If the corresponding escape nets are able to be detoured beyond its minimal routing contour in the left or right pin array to decrease the number of crossings in G, one new vertex representing the detoured nets will be separated from the original vertex in *V1* or *V2* and reassigned onto a routable location to eliminate the corresponding crossings. After reassigning the new vertex, the bipartite graph, *G*, will be modified. If there exists no crossing in *G*, the two-sided assignment process will stop and an initial routability-driven net order can be obtained according to the connection ordering of the ordered edges in *G*. For some nets including in a single edge, the inner ordering of the nets can be assigned and a final routability-driven net order can be further obtained. Refer to the final bipartite graph in Fig. 4(a), firstly, two escape nets, 14 and 6, are sequentially selected to be reassigned on the left component and two escape nets, 12 and 13, are selected to be reassigned on the right component, respectively. Furthermore, two escape nets, 1 and 4, are sequentially selected to be reassigned in the left component and one escape net, 10, is selected to be reassigned in the right component as illustrated in Fig. 4(b), respectively. Based on the inner net ordering, 8->7 and 12->13, for two groups of nets, (7, 8) and (12, 13), the final routability-driven net order can be obtained as 14->6->2->5->3->10->11->9->1->4->8->7->12->13 for simultaneous escape routing.

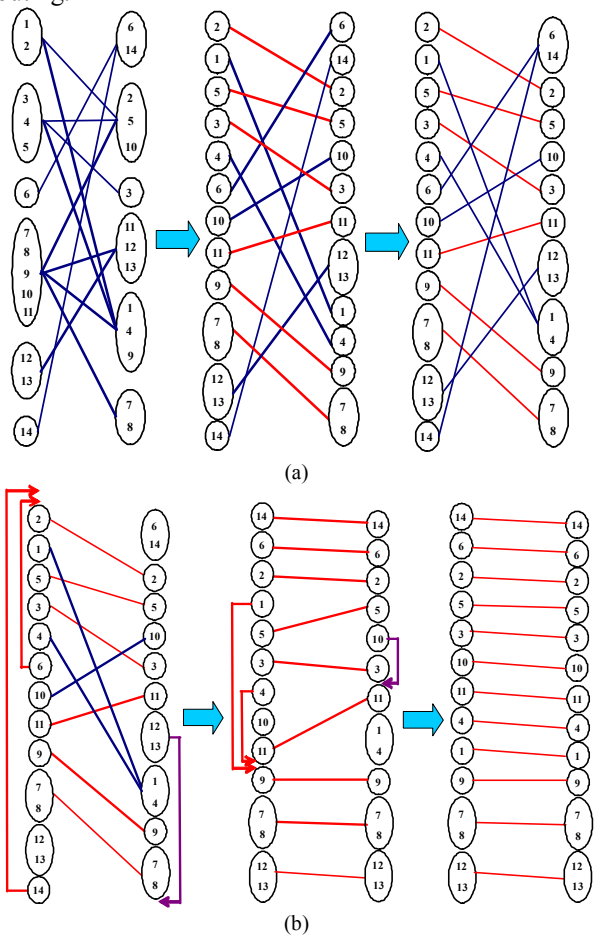


Fig. 4 Determination of routability-driven ordering for 14 escape nets

3.2 Transformation of Net Numbers

Given two pin arrays with n escape nets and the determined routability-driven net order, $\alpha(1)->\alpha(2)->\ldots->\alpha(n-1)->\alpha(n)$, it is known that $(\alpha(1), \alpha(2),\ldots, \alpha(n-1), \alpha(n))$ is a permutation of $(1, 2,\ldots, n-1, n)$. It is desired that the determined net order, $\alpha(1)->\alpha(2)->\ldots->\alpha(n-1)->\alpha(n)$, can be mapped into the order, $1->2->\ldots->n-1->n$, for ordered escape routing. Hence, the numbers of all the nets must be renumbered by using the replacement of $\alpha(i)$ with i, $1 \le i \le n$. It is important for the routing process to store the information of the renumbering transformation for the recovery of net numbers. Refer to the two pin arrays with 14 escape nets in Fig. 2, given the determined routability-driven net order, 14->6->2->5->3->10->11->9->1->4->8-

>7->12->13, of 14 escape nets, the net order can be mapped into 1->2->3->4->5->6->7->8->9->10->11->12->13->14 by using the replacement of all the net numbers in the transformation of net numbers as shown in Fig. 5.

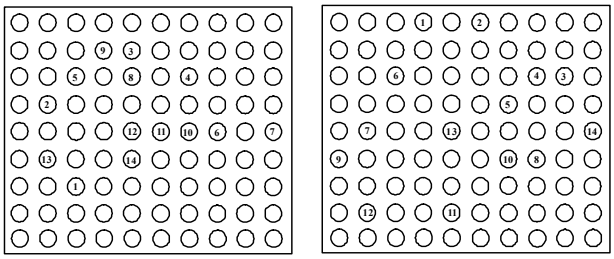


Fig. 5 Transformation of net numbers for 14 escape nets

3.3 Simultaneous Escape Global Routing

Since the determined net order, α(1)->α(2)->…->α(*n*-1)->α(*n*), is mapped into the order, 1->2->…->*n*-1->*n*, the simultaneous escape global routing problem can be treated as two one-sided ordered escape global routing problems. Given a pin array with *n* escape nets, the routing process for one-sided ordered escape global routing can divided into two sequential steps: *Routability-driven pin assignment* and *Global wire assignment*.

In routability-driven pin assignment, the routability-driven pins for one-sided ordered escape routing can be divided into *transfer pins* and *boundary pins*. Generally speaking, transfer pins are used to give the ordering transformation of the escape terminals for single-layer routing and boundary pins on the escape side are used to connect the escape nets. According to the assignment process of routability-driven pins in ordered escape routing[8], the transfer pins and the boundary pins on one escape side can be successfully assigned to avoid the crossing constraint in single-layer routing. Refer to the left pin array in Fig. 5, it is assumed that the pin array is rotated clockwise by 90-degree. If the capacity between two adjacent pins is 2, the number of available locations for boundary pins is 16. Clearly, the escape terminals, 1, 2, 9, 10 and 14, are relocated to avoid the crossing condition for single-layer routing by assigning 5 transfer pins. As illustrated in Fig. 6, 5 transfer pins and 14 boundary pins are assigned for all the escape nets.

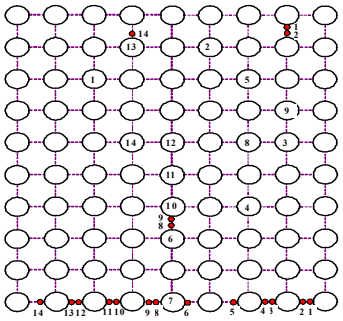
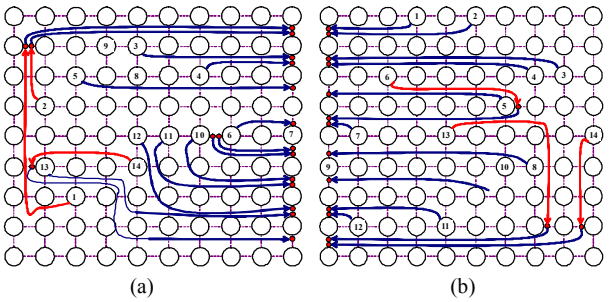


Fig. 6 Transfer and boundary pin assignment of 14 escape nets

In general, a single-layer routing plane with an array of pins can be treated as an array of routing cells. It is known that all the escape terminals, transfer pins and boundary pins must be fixed on the routing cells and all the connections from the escape terminals or the transfer pins to their boundary pins or from the escape terminals to their transfer pins must be routed in a single layer.

In global wire assignment, firstly, an undirected edge-weighted routing graph, $G(V, E)$, can be constructed as follows: every vertex in V represents a common boundary between two adjacent routing cells, each undirected edge in E represents a routing path between two boundaries inside any routing cell and the weight of any edge represents the number of the maximum paths between two boundaries. Furthermore, all the connections from the escape terminals or the transfer pins to their boundary pins must be sorted in an increasing order according to the distance between the routing pin and its boundary pin and the global wires of the sorted connections can be sequentially assigned by using straight routing paths. After assigning the global wires of the connections, the weights of the corresponding edges in the routing graph must be modified. Finally, all the connections from the escape terminals to their transfer pins must be sorted in an increasing order according to the Manhattan distances of all the connections. For any unassigned global connection, based on the edge-weighted routing graph, the global wire of the sorted unassigned connection can be obtained by finding the available shortest path from the vertex whose corresponding boundary is the nearest to its escape terminal to the vertex whose corresponding boundary is the nearest to its transfer pin. After assigning the global wire of the connection, the available capacity between any pair of two routing boundaries will be modified to maintain single-layer routing by assigning zero or decreasing by 1. Furthermore, the iterative assignment process for the other sorted connections will continue. Until the global wires of all the connections are assigned, the global wire assignment for one-sided ordered escape global routing will stop. Refer to the two pin arrays in Fig. 5, based on routability-driven pin assignment and global wire assignment in one-sided ordered escape global routing for the two pin arrays, the routing result for simultaneous escape global routing can be obtained as shown in Fig. 7.



(a) (b)

Fig. 7 Simultaneous escape global routing of 14 escape nets

3.4 Simultaneous Escape Detailed Routing

For the two pin arrays, simultaneous escape detailed routing can be further divided into two steps: *Crossing-point assignment* and *Physical path assignment*.

For the global wires of all the connections in two given pin arrays, the wire ordering on all the routing boundaries can be easily obtained to maintain the non-crossing constraint in a single layer. Based on the minimum-width and spacing rules, any routing boundary between two adjacent pins can be divided into some routing tracks. The feasible crossing points on the routing tracks must be further assigned for any global wire to satisfy the design rules. In crossing-point assignment, it is assumed that there are t tracks numbered from 1 to t between two adjacent bump balls. The global wires through the space between the two bump balls can be divided into a left set, N_L, a straight set, N_S, and a right set, N_R. Firstly, the crossing points of the global wires in N_L are assigned onto the available routing tracks from left to right. Furthermore, the crossing points of the global wires in N_R are assigned onto the available routing tracks from right to left. Finally, the crossing points of the global wires in N_S are assigned onto the remaining available routing tracks from left to right.

Based on the assignment result of the crossing-points in an array of routing cells, the endpoints of all the two-terminal connections inside a routing cell are located on the escape terminals, transfer pins, boundary pins or crossing-points and all the connections do not cross each other. In physical path assignment, all the two-endpoint connections inside a routing cell can be divided into *L-type connections* and *I-type connections*. Clearly, two endpoints in any *L-type* connection are located on two adjacent boundaries and two endpoints of any *I-type* connection are located on two opposite boundaries. Since all the connections inside a routing cell do not cross each other, all the L-type connections can be further divided into at most 4 covering groups according to the locations of two endpoints in an L-type connection. For any covering group, the physical paths of all the L-type connections are sequentially assigned by using L-type or stair-type paths in a diffusion-based style. After assigning the physical paths of all the L-type connections inside a routing cell, the physical paths of all the I-type connections inside a routing cell are sequentially assigned from left to right or from bottom to top by using I-type, Z-type or stair-type paths in a river routing algorithm[12]. As the physical paths of all the two-endpoint connections inside the routing cells are fully assigned for two given pin arrays, simultaneous escape detailed routing in two given pin arrays is completed. Refer to the routing results of the global wires in Fig. 7, by using crossing-point assignment and physical path assignment, the detailed routing result of all the escape nets between two given pin arrays can be obtained as shown in Fig. 8.

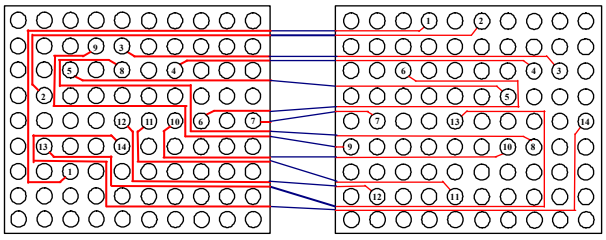


Fig. 8 Simultaneous escape detailed routing of 14 escape nets

3.5 Recovery of Net Numbers

As all the escape nets between two pin arrays are completely routed for simultaneous escape routing, the renamed net numbers in two pin arrays must be further recovered into the original net numbers according to the stored information in the transformation of net numbers. As a result, the final escape routing result between two pin arrays can be obtained for single-layer routing. Refer to the detailed routing result in Fig. 8, the final routing result of 14 escape nets between two pin arrays can be obtained for simultaneous escape routing by using the recovery of net numbers as shown in Fig.9.

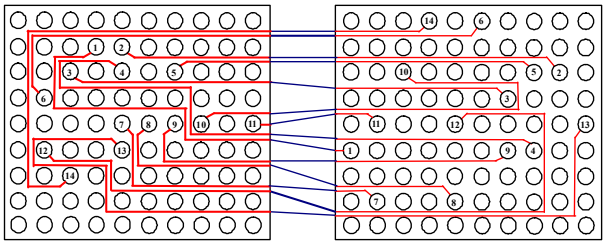


Fig. 9 Recovery of net numbers for 14 escape nets

3.6 Analysis of Time Complexity

Given two arrays of pxq and rxs pins with n escape nets, firstly, it is clear that the time complexity of determining a routability-driven net order is O($nlogn$). Based on the determined net order, the time complexity of transforming all the net numbers is O(n). For the analysis of time complexity in one-sided ordered escape global routing, based on the reassigned escape terminals and their reassigned locations in the determination of a routability-driven net order, the time complexity in *transfer pin assignment* is O(n). Furthermore, the time complexity in *boundary pin assignment* for all the escape nets is also O(n). Hence, the time complexity in the *routability-driven pin assignment* is O(n). Based on the non-crossing constraint in single-layer routing, the time complexity of constructing a routing graph is O(rs). Based on the ordering of the sorted nets, the time complexity in *global wire assignment* is O($nlogn$+rs). Hence, the time complexity in one-sided ordered escape global routing is O(n^2+rs). By using one-sided ordered escape routing for two pin arrays, it is clear that the time complexity in simultaneous global routing is O(n^2+pq+rs).

For the analysis of time complexity in simultaneous

escape detailed routing, the capacities between two adjacent pins in two pin arrays are constant and the number of two-endpoint connections inside a routing cell is constant. Hence, the time complexity in *crossing-point assignment* for all the routing cells in two pin arrays is O(*pq*+*rs*). Since the number of two-endpoint connections inside a routing cell is O(1), the time complexity in *physical path assignment* inside a routing cell is O(1). Hence, the time complexity in the *physical path assignment* for all the routing cells is O(*pq*+*rs*). Hence, the time complexity in simultaneous escape detailed routing is O(*pq*+*rs*). Finally, the time complexity of recovering all the net numbers is O(*n*). Based on the time complexities in five sequential phases, the time complexity of the proposed routing approach for simultaneous escape routing is O(n^2+*pq*+*rs*).

IV. EXPERIMENTAL RESULTS

For simultaneous escape routing, our proposed approach has been implemented by using standard C++ language and run on an Intel Core2 Quad/Q9450 2.66GHz machine with 2GB memory. Three tested examples, S01, S02 and S03, are from Kong's published paper[11]. The first smaller example, S01, has 58 escape nets between an array of 30x12 pins and an array of 30x7 pins. The second larger example, S02, has 62 escape nets between an array of 34x18 pins and an array of 34x18 pins. The third example, S03, is a 100-signal graphics processor unit(GPU) and has 100 escape nets between an array of 42x11 pins and an array of 52x2 pins. In Table I, "*LArray(#Cap)*" denotes a left array with a given capacity, "*RArray(#Cap)*" denotes the right array with a given capacity and "*#Net*" is the number of escape nets between two given pin arrays. Furthermore, we also implement Kong's flow-based approach[11] in single-layer routing to compare the final routing result and CPU time with our proposed approach for the tested examples.

In this experiment, the routability, *RRatio*, is defined as the routed rate of the escape nets in simultaneous escape routing. The experimental results in Table I show that our proposed routing approach and the flow-based approach can achieve 100% routability for simultaneous escape routing. Compared with Kong's O($(pq+rs)^3$) flow-based approach[11], it is known that our proposed approach has better time complexity in O(n^2+*pq*+*rs*). Besides that, the experimental results show that our proposed approach achieves 100% routability for the tested examples and reduces CPU time by 54.1% on the average. Clearly, the experimental results show that our proposed approach is more efficient than Kong's flow-based approach in single-layer routing.

V. CONCLUSIONS

Given a set of *n* escape nets between an array of *p*x*q* pins and an array of *r*x*s* pins, firstly, a routability-driven net order between two given pin arrays is determined for simultaneous escape routing. Furthermore, based on one-sided ordered escape global routing for two pin arrays, an efficient approach is proposed to solve the routing problem for simultaneous escape routing. Clearly, the experimental results show that our proposed approach is more efficient than Kong's flow-based approach in single-layer routing.

Table I Experimental results for simultaneous escape routing

	LArray (#Cap)	*RArray (#Cap)*	*#Net*	*Kong's Flow-based Approach[11]*		*Our Approach*	
				RRatio(%)	*Time(s)*	*RRatio(%)*	*Time(s)*
S01	*30x12(2)*	*30x7(2)*	*58*	*100%*	*0.23(100%)*	*100%*	*0.12(52.2%)*
S02	*34x18(2)*	*34x18(2)*	*62*	*100%*	*2.94(100%)*	*100%*	*1.22(41.5%)*
S03	*42x11(2)*	*52x2(1)*	*100*	*100%*	*4.72(100%)*	*100%*	*2.07(43.9%)*

REFERENCES

1 Y. Birk and J. B. Lotspiech, "On finding non-intersecting straightline connections of grid points to boundary", *Journal of Algorithms*, Vol.13, No.4, pp.636-656, 1992.

2 W. T. Chan and F. Y. L. Chin, "Efficient algorithms for finding maximum number of disjoint paths in grids," *SODA*, pp.454-463, 1997.

3 J. W. Fang, I. J. Lin, P. H. Yuh, Y. W. Chang, and J. H. Wang, "A routing algorithm for flip-chip design," *IEEE International Conference on Computer-Aided Design*, pp. 753–758, 2005.

4 Y. Tomioka and A. Takahashi, "Monotonic parallel and orthogonal routing for single-layer ball grid array packages," *Asia South-Pacific Design Automation Conference*, pp.642–647, 2006.

5 J. W. Fang, C. H. Hsu and Y. W. Chang, "An integer linear programming based routing algorithm for flip-chip design," *Design automation Conference*, pp. 606–611, 2007.

6 L. Luo and D. F. Wong, "Ordered escape routing based on Boolean satisfiability," *Asia South-Pacific Design Automation Conference*, pp.244–249, 2008.

7 L. Luo and D. F. Wong, "On using SAT to ordered escape problem," *Asia South-Pacific Design Automation Conference*, pp.594-599, 2009.

8 J. T. Yan, C. W. Ke and Z. W. Chen, "Ordered escape routing via routability-driven pin assignment," *ACM Great Lakes Symposium on VLSI,* pp.417-422, 2010.

9 M. M. Ozdal and D. F. Wong, "Simultaneous escape routing and layer assignment for dense PCBs," *IEEE/ACM International Conference on Computer-Aided Design*, pp.822-829, 2004.

10 T. Yan and D. F. Wong, "A correct network-flow model for escape routing," *Design Automation Conference*, pp.332-335, 2009.

11 H. Kong, T. Yan and D. F. Wong, "Optimal simultaneous pin assignment and escape routing for dense PCBs," *Asia South-Pacific Design Automation Conference*, pp.275–280, 2010.

12 Q. Ma, T. Yan and D. F. Wong, "A negotiated congestion based router for simultaneous escape routing," *International Symposium on Quality Electronic Design*, pp.606-610, 2010.

13 L. Luo, T. Yan, Q. Ma, D. F. Wong and T. Shibuya, "B-escape: a simultaneous escape routing algorithm based on boundary routing," *International Symposium on Physical Design*, pp.19-25, 2010.

14 C. P. Hsu, "General river routing algorithm," *Design Automation Conference*, pp.578-583, 1983.