Substrate Topological Routing for High-Density Packages

Shenghua Liu, *Student Member, IEEE*, Guoqiang Chen, Tom Tong Jing, *Member, IEEE*, Lei He, *Senior Member, IEEE*, Tianpei Zhang, Robi Dutta, and Xian-Long Hong, *Fellow, IEEE*

***Abstract*—Off-chip substrate routing for high-density packages is on the critical path for time to market. Compared with on-chip routers, existing commercial tools for off-chip routing have lower routability and often result in a large number of unrouted nets for manual routing. In this paper, we explain why planar routing is still required with multiple routing layers for substrate routing and then propose a flexible via-staggering technique to improve routability. In addition, we develop an efficient yet effective substrate routing algorithm, applying dynamic pushing to tackle the net ordering problem and reordering and rerouting to further reduce wire length and congestion. Compared with an industrial design tool that leaves 936 nets unrouted for nine industrial designs with a total of 6100 nets, our algorithm reduces the unrouted nets to 212, a 4.5-times net number reduction, which translates to design time reduction.**

***Index Terms*—Integrated circuit (IC) package, routability, substrate, system-in-package, topological routing.**

I. Introduction

An IC PACKAGE (see Fig. 1) usually uses a *ball grid array (BGA)* substrate and *wire bonding* or *flip-chip* to connect a die to the substrate. However, high-density integration inside the package makes off-chip routing a challenging task. For a wire-bonding die, the I/O pads of the die are connected to the *bond pads* around the cavity through bonding wires. For a flip-chip die, on-chip *redistribution layer* routing [1], [2] first connects the I/O pads to *bump pads*, and *escape routing* [3]–[8] then breaks out bump pads to the boundary of the die at the *escape break points* in the buildup or signal layers. Finally, *substrate routing* connects escape break points of flip-chip dies or bond pads of wire-bonding dies to *solder balls* (usually in the bottom layer) of a BGA package substrate.

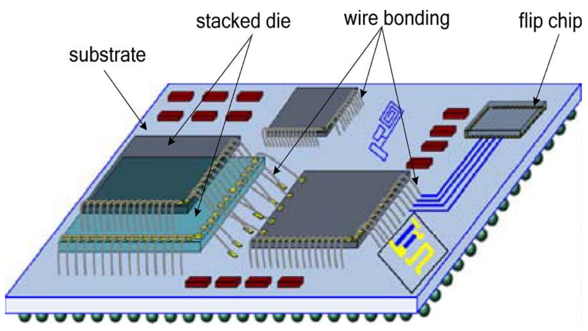


Fig. 1. Example of an IC package.

Substrate routing can be divided into two steps: 1) *topological routing* and 2) *detailed routing*. While detailed routing is discussed in [9]–[11], this paper studies topological routing. We justify why planar routing is still required when multiple routing layers are available for substrate routing and then propose a flexible via-staggering technique to improve routability. In addition, we develop an efficient yet effective substrate routing algorithm, applying dynamic pushing to tackle the net ordering problem and reordering and rerouting to further reduce wire length and congestion. Compared with an industrial design tool that leaves 936 nets unrouted for nine industrial designs with a total of 6100 nets, our algorithm reduces the unrouted nets to 212, a 4.5-times net number reduction. The average wire length reduction by our method is 13.9%.

The rest of this paper is organized as follows: Section II reviews existing work and formulates a new routing problem. Section III describes our algorithms in detail. Section IV presents experimental results, and Section V concludes this paper.

II. Background and Problem Formulation

A. Comparison With the Existing Work

Advanced packaging technology such as system-in-package needs flexible locations for bond pads and escape break points, which are called *start points* of nets in this paper. However, most existing work in topological routing [12]–[14] assumes that start points are located side by side with respect to solder balls. Our routing algorithm to be presented does not suffer from such location constraints.

Existing substrate routing work [15], [16] does not specify the destination ball to a start point in substrate routing,

Manuscript received April 18, 2008; revised July 7, 2008. Current version published January 21, 2009. This work was supported in part by the National Science Foundation under CAREER Award CCR-0093273 and in part by the National Natural Science Foundation of China under Grant 60720106003 and 90607001. This paper was recommended by Associate Editor L. Scheffer.

S. Liu and X.-L. Hong are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: liuch05@mails.tsinghua.edu.cn; hxl-dcs@tsinghua.edu.cn).

G. Chen and R. Dutta are with Magma Design Automation, Inc., San Jose, CA 95110 USA (e-mail: gchen@Magma-DA.COM; Robidutta@aol.com).

T. T. Jing was with the Department of Electrical Engineering, University of California, Los Angeles (UCLA), Los Angeles, CA 90095 USA. He is now with Synopsys, Inc., Mountain View, CA 94043 USA (e-mail: jingtong_eda@hotmail.com).

L. He is with the Department of Electrical Engineering, UCLA, Los Angeles, CA 90095 USA (e-mail: lhe@ee.ucla.edu).

T. Zhang was with the University of Minnesota, Minneapolis, MN, 55455 USA. He is now with Cadence Design Systems, San Jose, CA 95134 USA (e-mail: dune.ocean@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2008.2009154

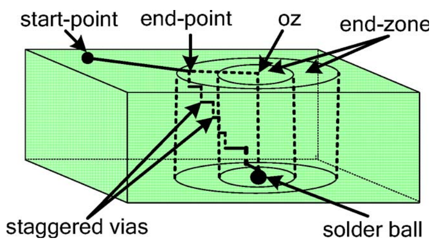


Fig. 2. End zone and flexible endpoint.

which makes routing easier as there is no constraint on net ordering. However, printed circuit board designers prefer to specify solder ball assignment as a design constraint. Our routing algorithm can deal with the case of specified solder ball assignment.

When dropping signal vias, the vias need to be staggered and close to the locations above assigned destination balls. In substrate routing, which is preferred to be planar (as will be discussed in Section II-B), nets are finally connected to the solder balls in the bottom layer by staggered vias, where vias crossing multiple layers cannot be stacked exactly one on top of the other due to the required offset, called minimum and maximum staggered via pitches. The former is determined by via manufacturing technology. The latter is determined by the power/ground (P/G) network since the pitch should not impact the integrity of the P/G plane. Thus, for example, taking a typical four-two-four package,[1] we have the flexibility to decide where the planar routing ends and the staggered via starts. We define the zone where the staggered via can start as the *end zone*. The end zone includes two circles (see Fig. 2). The radii of the two circles are d_1 and d_2, respectively, where $d_1 = \sum_i md_i$ and $d_2 = \sum_i pd_i$, with md_i and pd_i being the minimal and maximal staggered via pitches in the layer with index i, respectively. However, one problematic simplification by most existing work [12], [14] is to ignore the staggered via requirement and route to a fixed endpoint. In [13], via assignment was performed ahead of routing, but it is suitable only for the case of fixed via location with two-layer packages. This paper performs via staggering so that a net can be connected to any point in its end zone defined by its solder ball assignment. Compared with connecting to a fixed point inside the end zone as in the existing work, it improves routability.

B. Need of Planar Routing

In practice, most wire-bonding packages use only one signal routing layer due to the cost constraint. It is more common to have multiple signal routing layers for flip-chip packages. Although there are multiple routing layers for flip-chip packages, planar routing is still needed. First, signal via location is limited within the end zone, which, in turn, is decided by solder ball assignment and staggered via manufacturing technology. In other words, the signal vias are allowed in very limited locations due to manufacturability, leaving space for P/G vias.

[1]It is a ten-layer package. From top to bottom, there are four buildup layers, two core layers, and four buildup layers. It can be used as GND–signal–Vdd–signal–GND–Vdd–signal–GND–signal–Vdd.

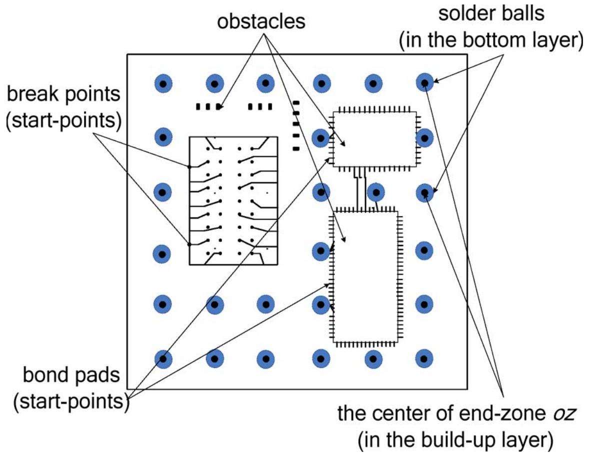


Fig. 3. SRG in a buildup layer.

Second, substrate routing often does not allow vertical detours as such detours introduce extra vias, which may destroy the signal integrity for high-speed differential signals. Therefore, multilayer routing formulation such as that for on-chip routing does not improve routability given the aforementioned two via constraints.

C. Problem Formulation

In this paper, we assume that escape routing has decided the location and layer of each start point. The key to the topological routing problem is single-layer routing, which is performed on the substrate routing graph (SRG) (see Fig. 3) that maps the start points, end zones, and dies (as obstacles) on a graph. The problem of substrate topological routing for package is formulated as follows.

Formulation 1 (Substrate Topological Routing): Given start points, solder balls in the bottom layer, netlist defined by solder ball assignment, and obstacles (including the escape area for escape routing, prerouted connections, vias, and other obstacles in the layer), find a topological routing solution connecting each start point to any point in the end zone for its assigned solder ball, such that the routed nets inside the obstacles are planar, satisfying the capacity constraints, and have minimal wire length.

In our routing algorithm, the SRG in a buildup layer is further discretized by a set of simple elements such as triangles or quadrilaterals in two dimensions. There are high-density and aligned start points, prerouted connections, and all kinds of possible polygons on the SRG plane. Considering these practical constraints in this paper, we apply a triangle mesh by constraint Delaunay triangulation (CDT) [17], which guarantees a low computational cost and reasonably well-shaped elements. Uniformly spreading points, called *particles* U, are added in the same way as [18] to the SRG plane for particle-insertion-based CDT (PCDT) construction. Then, we build a PCDT graph based on start points, the centers of the end zones (oz's), the obstacles, the particles U, and the boundary of the SRG plane. Thus, the start points and the oz's become vertices of triangles. Finally, a dual graph of PCDT is accordingly built,

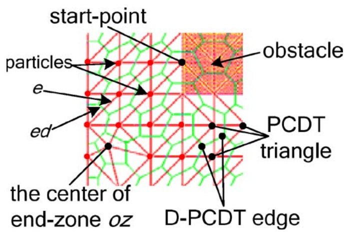


Fig. 4. Partial PCDT and D-PCDT.

```
FOR each build-up layer of signal routing in the package, from top to bottom,
DO
{
  s1: Initialization
  {
    s1.1: Generate PCDT graph and its dual graph D-PCDT;
    s1.2: Determine end-zone for each net, considering via assignment;
    s1.3: Determine initial net order such that the net with shortest distance
          between the start-point and end-zone is routed first;
    s1.4: Set congestion threshold $\psi := \psi_0$;
    s1.5: Set bound of pushes for a single net $\xi := +\infty$;
    s1.6: Set status to be initial routing;
  }
  s2: WHILE( !($\psi <= \psi^*$ && $\xi == 0$) )
  {
    s2.1: FOR all the nets in current order
    DO
    {
      s2.1.1: IF(status is rerouting) rip up the net;
      s2.1.2: Perform $A^*$-based dynamic search $ADS^*(\psi, \xi)$ for the net path;
      s2.1.3: Perform post-optimization only for the pushed nets;
    }
    s2.2: IF(no more wire length reduction)
    {
      s2.2.1: IF($\psi > \psi^*$) decrease $\psi$;
      s2.2.2: IF($\xi > 0$) decrease $\xi$;
    }
    s2.3: Reorder nets based on strategies of whole reordering
          and partial reordering for wire length reduction;
    s2.4: Set status to be rerouting;
  }
}
```

Fig. 5. Algorithm overview.

which we name D-PCDT. An example of PCDT and its dual-graph D-PCDT are shown in Fig. 4.

III. Algorithms

This section first introduces the algorithm overview and then discusses the core algorithms in detail, namely, A^*-based[2] dynamic searching algorithm (ADS^*) and reordering algorithm.

A. Algorithm Overview

The overview of our topological routing algorithm is shown in Fig. 5. There are two core algorithms: 1) ADS^* and 2) reordering algorithm. The two algorithms are carried out layer by layer from top to bottom. Initial routing and rerouting both employ the ADS^* algorithm.

As shown in Fig. 5, one net is ripped up at a time, and all nets are ripped up and rerouted in every iteration, even if some of the nets do not pass through a congested area. All the nets in substrate routing have exactly two pins. ADS^* is developed to route net by net in s2.1.2 in Fig. 5. During the process of routing, two kinds of unnecessary detours may happen. Therefore, post optimization is performed in the algorithm as s2.1.3 in Fig. 5. One is *cross-twice-detour*, as shown in Fig. 6(a), where a path passes across one PCDT edge twice and the two cross points on the edge are neighbors. Then, this detour can be removed, as shown in Fig. 6(a). The other is *cross-all-edge-detour*, as shown in Fig. 6(b), where a path continuously passes across three edges of the same triangle. Then, it can be optimized, as shown in Fig. 6(b). Then, in s2.3, before the next routing iteration, we reorder all the nets for the goal of bent wire reduction (as will be discussed in Section III-C).

The *bound of pushes for a single net* ξ is given as a threshold in each iteration to control pushing. ξ starts at infinity, as shown in s1.5 in Fig. 5, and is gradually reduced to 0, as shown in s2.2.2 in Fig. 5.

[2]A^* [19] is an efficient graph/tree search algorithm that finds a path from a given initial node to a given goal node. Compared with other typical algorithms such as Lee's maze [20], Moore's algorithm D [21], and Dijkstra's algorithm, the A^* algorithm is more general and more often has lower time complexity because of its heuristic estimated cost.

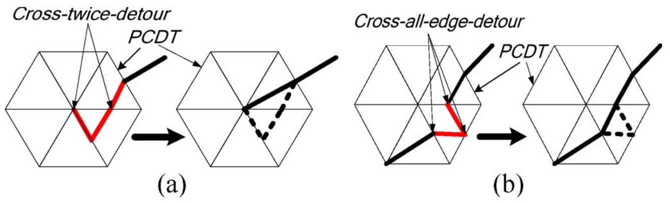


Fig. 6. (a) Cross-twice-detour. (b) Cross-all-edge-detour.

B. ADS^*

To route net by net, we develop a searching algorithm, called ADS^*, as shown in Fig. 7. For each two-pin net, ADS^* finds the shortest path on the D-PCDT graph (on the PCDT graph, it is a path from one triangle to another triangle) subject to a capacity constraint. The capacity C_{ed} of each D-PCDT edge ed can be calculated as follows. Let e be the edge of PCDT and e crosses edge ed. If edge e is inside an obstacle, on the boundary of the obstacle, or on the boundary of the SRG plane, then $C_{ed} = 0$. Otherwise, $C_{ed} = l_e$, where l_e is the length of edge e. The congestion of edge ed is defined as follows.

If $C_{ed} = 0$, edge ed cannot have any net passing along it. Hence, $\eta_{ed} = +\infty$ is defined for path searching. Otherwise

$$\eta_{ed} = \frac{\sum_i (w_i + s_i)}{C_{ed}} \tag{1}$$

where w_i and s_i, respectively, are the wire width and space of net i passing through edge e or along edge ed.

The ADS^* search algorithm is based on the framework of heap-implemented A^* algorithm. For the substrate routing problem, two key technologies, namely, *dynamic pushing* and *flexible via staggering*, are embedded into the procedure and the *evaluation function* (i.e., the sum of actual and estimated costs) of the A^* algorithm. Then, the evaluation function can guide the search to *push* or *detour* the routed nets blocking the current net that is searching its path. Instead of finding the least-cost path from the source node to the destination node in the A^* algorithm, ADS^* aims to find the least-cost path from the source node to a zone. Once the path on the D-PCDT graph

```
s1: Initialization
{
  s1.1: Create a heap H;
  s1.2: Calculate evaluation function for neighbor
        triangles of start-point by Equations (2), (3), and (8);
  s1.3: Push neighbor triangles into the heap H, set the pushed nets list
        of each triangle into null, and flag them as visited;
}
s2: WHILE(the heap H is not empty)
{
  s2.1: Find the triangle i with minimal evaluation value in the heap H;
  s2.2: Let triangle i be search frontier;
  s2.3: Delete triangle i from the heap H;
  s2.4: IF(searching should stop at triangle i by rules R1 and R2)
    s2.4.1: break;
  s2.5: Choose neighboring triangle frontier*s of triangle i satisfying
  {
    s2.5.1: Triangle i is not visited;
    s2.5.2: Congestion value $\eta$ <= $\psi$ ;
    s2.5.3: Number of pushes by the net in triangle i <= $\xi$;
  }
  s2.6: For each frontier* f* in frontier*s
  {
    s2.6.1: Assign the carried nets list of frontier to f*;
    s2.6.2: If there is any net $n_b$ blocking the current net going from frontier to f*
              add it into the pushed nets list of f*
              (pushing operation);
    s2.6.3: Calculate evaluation function for f* by Equations (5), (6), (8), and (9),
            considering the cost of pushing/detouring and end-zone;
  }
  s2.7: Push selected neighbor triangle frontier*s into the heap H
        and flag them as visited;
}
```

Fig. 7. Algorithm $ADS^*(\psi, \xi)$.

is found, cross points on the PCDT graph are simultaneously assigned. We next discuss the key techniques of ADS^*.

To reduce congestion, we use ψ as a *congestion threshold* to guide ADS^*. Highly congested edges with higher *congestion values* η_{ed}, as discussed earlier in this section, are forbidden from being passed. That is, triangles with those highly congested edges that the net in the $frontier$ needs to pass through cannot be a $frontier^*$ during search, as shown in s2.5.2 in Fig. 7, and cannot be pushed into the heap. ψ starts with a large number as ψ_0 and gradually decreases to the target congestion threshold ψ^*. ψ^* is an empirical value between 0.8 and 1.0. s.2.6 in Fig. 7 gives the procedure of pushing blocked nets and the calculation of the evaluation function, considering the cost of pushing/detouring and the end zone.

1) Searching With Dynamic Pushing: To tackle the net ordering problem, we embed a *dynamic pushing* technique during the net path search. Fig. 8 illustrates how dynamic pushing works. Nets A and B in Fig. 8(a) are symmetric, and all possible net orders are (A, B)–C (i.e., A–B–C and B–A–C), A–C–B, and C–(A, B). Only the order (A, B)–C can directly be solved by a traditional router such as the rubber-band method [22] shown in Fig. 8(b). However, dynamic pushing can handle net orders that the existing work cannot consider. For the net order A–C–B, after the ADS^* algorithm connects nets A and C, net B *pushes* routed net C and uses either path 1 or path 2, as shown in Fig. 8(c). For the net order C–(A, B), after the ADS^* algorithm connects net C, net A *pushes* routed net C, and then, net B also *pushes* net C and uses either path 1 or path 2, as shown in Fig. 8(d).

To choose path 1 or path 2, we need to evaluate the cost calculated by the evaluation function as ADS^* searches a net path. A heap is created in the initialization, as illustrated in s1.1

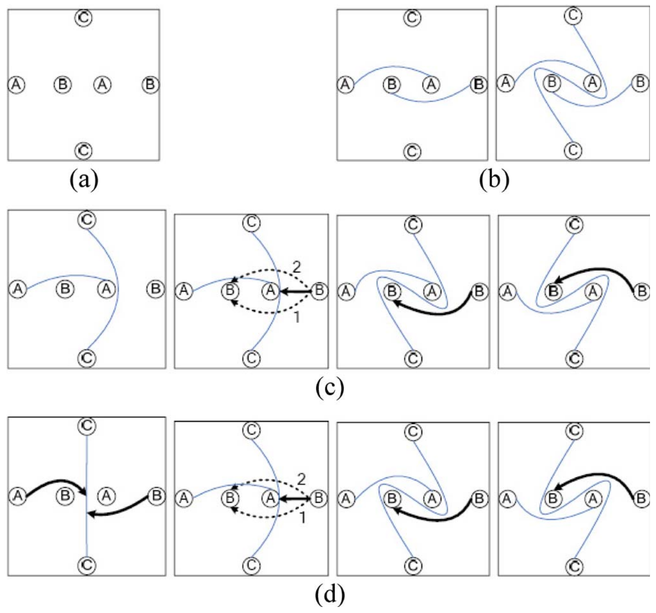


Fig. 8. Example of net ordering. (a) Nets initial location. (b) Net order (A, B)–C and routing. (c) Net order A–C–B and routing. (d) Net order C–(A, B) and routing. Only this work can obtain routing (c) and (d).

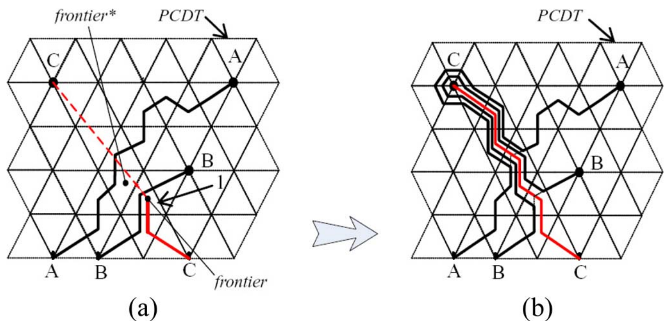


Fig. 9. Cost function calculation.

in Fig. 7. The evaluation function calculates the cost as the sum of actual and estimated costs. The search $frontier$ and other elements as candidates of the next $frontier$ in the heap keep their own records of the pushed nets list, the width w of the $frontier$, the actual cost rc, and the estimated cost ec. Before searching starts, the w, rc, and ec of the triangles incident with the start point are calculated, respectively, as follows:

$$w = w_0 + s_0 \tag{2}$$

$$rc = p_0 \times w \tag{3}$$

$$ec = h_0 \times w \tag{4}$$

where w_0 and s_0 are the wire width and space of net C, respectively; p_0 is the distance from the start point to the barycenter of an incident triangle; and h_0 is the distance from the barycenter to the endpoint. Then, those triangles are pushed into the heap as the candidates of the search $frontier$ and flagged as "visited." As shown in Fig. 9(a), when starting from the start point of net C, the w, rc, and ec in incident triangles are calculated.

Afterward, the candidate with the minimal cost in the heap is selected as the search $frontier$. The unvisited and uncongested neighbors adjacent to the search $frontier$ are selected to evaluate their cost. Then, the selected neighbors are pushed into the heap and flagged as "visited." We denote those selected neighbors of the $frontier$ as $frontier^*$'s and record the related w^*, rc^*, and ec^*. Let i be the ith net that blocks the movement from the search $frontier$ to a $frontier^*$. The wire width and space of net i are w_i and s_i, respectively. Then, the $frontier^*$ can recursively be calculated by the following formulas:

$$w^* = w + 2\sum_i (w_i + s_i) \tag{5}$$

$$rc^* = rc + \Delta p \times w^* \tag{6}$$

$$ec^* = h \times w^* \tag{7}$$

where Δp is the distance between the barycenter of the $frontier$ triangle and the $frontier^*$ triangle, and h is the distance between the $frontier^*$ triangle barycenter and the endpoint.

We explain the case shown in Fig. 9(a) with (5)–(7). Net B blocks net C when net C moves from the $frontier$ in triangle-1 to the $frontier^*$. If net C pushes net B, the $frontier^*$ contains two segments from each pushed net. Thus, the width of the search $frontier$ w^* adds double width and space of net B, as shown in (5), and results in the increase of the actual cost rc^* and the estimated cost ec^*. Otherwise, if net C detours around net B, although w^* is still w, the sum of h and Δp increase, as shown in (6) and (7). It is a tradeoff between pushing net B and detouring net B in the presence of the evaluation function. However, our dynamic programming technique and heap implementation in ADS^* give the optimal choice. In ADS^*, dynamic programming is used. In each searching step, the optimality of the subproblem is guaranteed by always choosing the first element from the minimum heap. Thus, the optimality of the original problem is guaranteed based on our evaluation function.

Equations (5)–(7) show the basic difference between conventional rip-up and reroute and dynamic pushing. Once a net is pushed, its width is added to w^*, and the cost computed in (6) and (7) contains the accumulated widths of all the nets pushed. However, the cost in conventional rip-up and reroute only contains the routing and detouring wire length.

Once a candidate with the minimal evaluation function value is selected from the heap, it recursively becomes the search $frontier$ for subsequent searching. Fig. 9(b) is an example of a routing solution found by using dynamic pushing with evaluation function.

Based on the aforementioned searching process with the evaluation function combining with dynamic programming (as illustrated in Fig. 7), ADS^* can find the optimal path between path 1 and path 2 in Fig. 8(c) or (d).

A four-step detailed router, i.e., Mighty, was presented in [23]. The core algorithms are Lee's maze [20] and post optimization. Three push techniques, namely, *unitpush, jumppush*, and *pointpush* [23], were proposed with some examples shown in [23, Figs. 3(k) and (l) and 4(a)–(c)] in the third step, i.e., weak modification. The essential difference between our dynamic pushing and push techniques in [23] is that [23] is based on two-layer routing technologies, whereas ours is based on planar routing. Therefore, our dynamic pushing is a wire push, whereas the push technique in [23] is only a via relocation. In [23, Figs. 3(k) and (l) and 4(a)–(c)], we can see that it does not push any wire at all. In addition, the method does not work if the route is only in one layer.

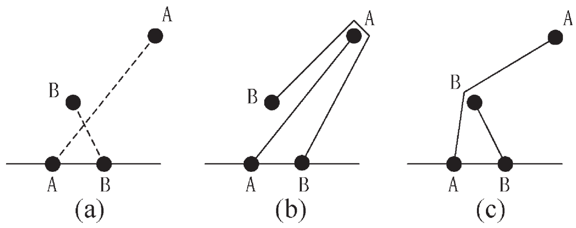


Fig. 10. Different routing results between the "traditional rip-up and reroute" and "dynamic pushing." (a) Given nets. (b) Routing result by the traditional rip-up and reroute. (c) Routing result by dynamic pushing.

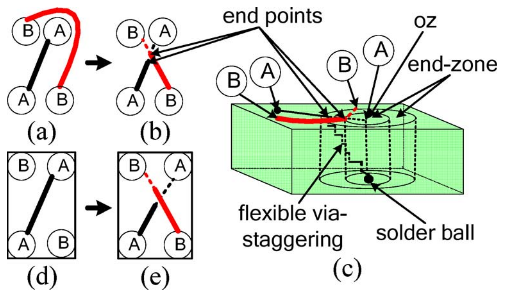


Fig. 11. Comparison between fixed end-point [(a) Detour and (d) Unroutable] and flexible via-staggering [(b) Nondetour, (c) 3-D description, and (e) Routable].

It is worth mentioning that "dynamic pushing" in (planar) substrate routing is different from the traditional rip-up and reroute that needs the proper net order for rerouting. This is illustrated in Fig. 10, where Fig. 10(a) shows the given nets A and B to be routed. Fig. 10(b) shows the routing result by the traditional rip-up and reroute if the net order is net A $\rightarrow$ net B. Without the proper net order (net B $\rightarrow$ net A), the routing result cannot be improved. Fig. 10(c) shows the routing result by dynamic pushing, which is independent of the net order. In addition, dynamic pushing leads to shorter wire length compared with the current best known method of topological routing (BKM) (as discussed in Section IV).

2) Searching With Flexible via Staggering: We use the *flexible via-staggering* technique for planar routing to stop at any point in the end zone. By considering this, ADS^* can reduce wire length and improve its routability. The following example shows the two advantages of the flexible via-staggering technique.

As shown in Fig. 11(a), in the traditional fixed endpoint case, net B must detour around routed net A to complete connection. However, the flexible via-staggering technique of ADS^* can both successfully route net B and obtain shorter wire length by changing the endpoint of net A when the endpoint is flexible, as shown in Fig. 11(b). Fig. 11(c) gives the 3-D description of the flexible via-staggering technique. Fig. 11(d) shows a worse case, where routed net A blocks all the possible routes for net B. In this case, the flexible via-staggering technique can also successfully route net B and obtain shorter wire length by changing the endpoint of net A, as shown in Fig. 11(e). Therefore, the

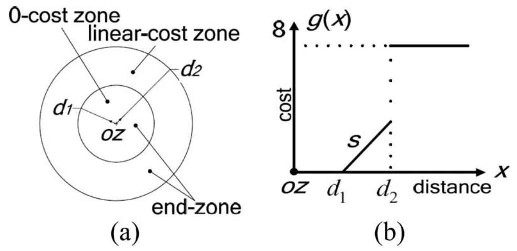


Fig. 12. (a) End zone. (b) End-zone cost.

flexible via-staggering technique can obtain higher routability and shorter wire length. Moreover, it has no extra expense since it makes use of the required via-staggering pitch.

The size of the end zone is decided by the minimum and maximum via-staggering pitches in the layers between the current buildup layer and the bottom layer, as discussed in Section II-A. Our search considers the cost caused by ending close to the oz as follows. The end zone can be viewed as the combination of two zones [see Fig. 12(a)]. We define the zone with a radius of d_1 as the 0-cost zone, whereas the zone within the circle with a radius of d_2 and outside the circle with a radius of d_1 is defined as the linear-cost zone.

In design, the minimum staggered via pitch is the best choice for the connection from the endpoint to the solder ball. Meanwhile, in the 0-cost zone, mostly, the endpoint can be connected to the assigned solder ball by using the minimum via-staggering pitch in a zigzag manner. We do not give any penalty for such connections. Therefore, we choose the cost of each position in the 0-cost zone to 0. However, while an endpoint is in the linear-cost zone, which means that the offset larger than the minimum via-staggering pitch has to be used, we need to give a penalty to reduce the negative effect on the integrity of the P/G plane. Thus, we define a linear cost function $g(x)$ for the cost of each location in the linear zone, as shown in Fig. 12(b), where s is the slope of $g(x)$. Based on parameters s, d_1, and d_2, we have the following theorem to derive the rules R1 and R2 for the searching algorithm ADS^*, where w is the width of the search $frontier$.

Theorem 1: R1: Every search $frontier$ ends no later than it enters the 0-cost zone.

R2: After the search $frontier$ enters the end zone, in every consecutive searching step, if $w > s$, searching ends; else, searching continues.

R1 and R2 extend the traditional search that finds the shortest path between two fixed locations to finding the shortest path from a fixed location to a zone. Since these rules determine the searching end, i.e., the endpoint, the estimated cost determined by the estimated endpoint location should also be impacted. Thus, (4) and (7) can be rewritten as follows. Note that based on R1, the distance h between the $frontier$ triangle barycenter to the oz satisfies $h \geq d1$. Thus

$$ec = \begin{cases} (h_0 - d_2) \times w + (d_2 - d_1) & \\ \quad \times \min(s, w), & h_0 > d_2 \\ (h_0 - d_1) \times \min(s, w), & d_1 \leq h_0 \leq d_2 \end{cases} \tag{8}$$

$$ec^* = \begin{cases} (h - d_2) \times w^* + (d_2 - d_1) & \\ \quad \times \min(s, w^*), & h > d_2 \\ (h - d_1) \times \min(s, w^*), & d_1 \leq h \leq d_2. \end{cases} \tag{9}$$

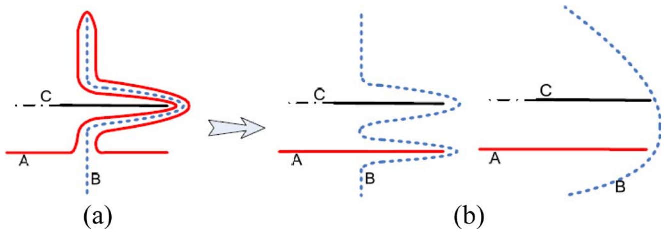


Fig. 13. Reordering and finding a better solution. (a) Net A is pushed twice. (b) Better solution.

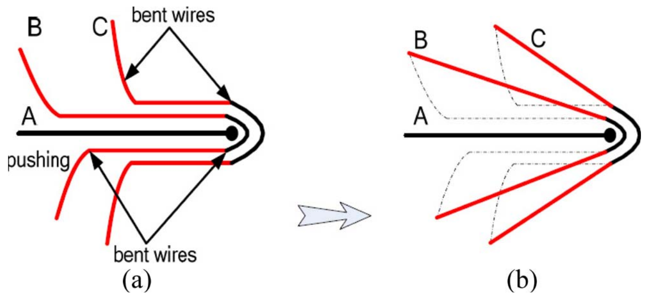


Fig. 14. (a) Bent. (b) Stretch.

It is worth mentioning that each net has its own start point and specified solder ball in the bottom layer. In addition, each solder ball has its own end zone in the related buildup layer. Each end zone has some PCDT triangles and D-PCDT edges inside. Thus, there is no conflict of endpoints inside the end zone between any two nets, as shown in Fig. 11(c). That is, any two nets unlikely end their routes at the same endpoint inside the end zone.

C. Reordering Strategy

Based on our experiments, we observe that routing nets frequently results in *bent wires* caused by pushing, as demonstrated in Figs. 13(a) and 14(a). *Bent wires* usually involve unnecessary detours and increase total wire length. We solve this problem by using rip-up and rerouting based on the following reordering strategies. That is, after one iteration of all nets routing by ADS^* (see s2.1 in Fig. 5), we reorder all nets (see s2.3 in Fig. 5). Then, we rip-up one net, leaving others still routed, and reroute it by $ADS^*(\psi, \xi)$ (see s2.1.1 and s2.1.2 in Fig. 5).

Our reordering strategy is as follows. In the first iteration of planar routing (see s1.3 in Fig. 5), short nets have a lower probability of blocking unrouted nets. Therefore, we order nets such that the net with the shortest distance between the start point and the end zone is first routed. In the later iterations, two strategies, i.e., *whole reordering* and *partial reordering*, are combined for reordering in each iteration.

We define a *net length ratio* $\delta = l/u$, where l is the net length acquired from the latest routing iteration, and u is the distance between the start point and the oz (the center of end zone). We first use *whole reordering*, i.e., *larger* δ *first*, to generate a new net order, which prioritizes nets with larger increases caused by *pushing*. For example, in Fig. 13(a), net A is first pushed by net B and then by net C. Thus, nets such as net A are allocated a high priority to allow them to "stretch." In Fig. 13(b), if the

TABLE I
TEST CASES AND EXPERIMENTAL RESULTS

Name of circuit	Package type	Package size*	Die(s) size*	Number of nets	Number of failed nets		Average wire length (mm)		Runtime (s)	
					BKM	new	BKM	new	BKM	new
case 1	2-2-2	35000 × 35000	14000 × 15000	474	52	16	9.10	8.49	2.06	11.33
case 2	2-2-2	30000 × 30000	9000 × 10500	543	10	0	8.24	7.99	1.90	5.34
case 3	3-1-3	40000 × 40000	9300 × 9300	800	306	49	1.65	1.37	13.81	16.86
case 4	3-2-3	35000 × 35000	12000 × 12000	506	82	6	22.90	17.80	6.90	8.46
case 5	3-2-3	40000 × 40000	20000 × 22000	891	98	45	5.93	5.27	2.60	13.65
case 6	4-2-4	40000 × 40000	20000 × 23000	990	198	51	9.10	7.93	9.95	26.62
case 7	4-2-4	45000 × 45000	20000 × 19000	1009	100	14	23.10	18.90	5.91	26.14
case 8	1-0-1	12000 × 12000	3900 × 6700 4400 × 5700 3200 × 4400	349	60	22	1.83	1.73	15.24	1.51
case 9	2-2-2	37500 × 37500	11000 × 10000 4700 × 3800 4600 × 5500	538	30	9	9.78	9.38	95.17	110.93
total	—	—	—	6100	936	212	—	—	—	—
average	—	—	—	—	—	—	10.20	8.76 (-13.9%)	—	—

(*: Package size and Die(s) size are given by width × length (μm) in rectangle.)

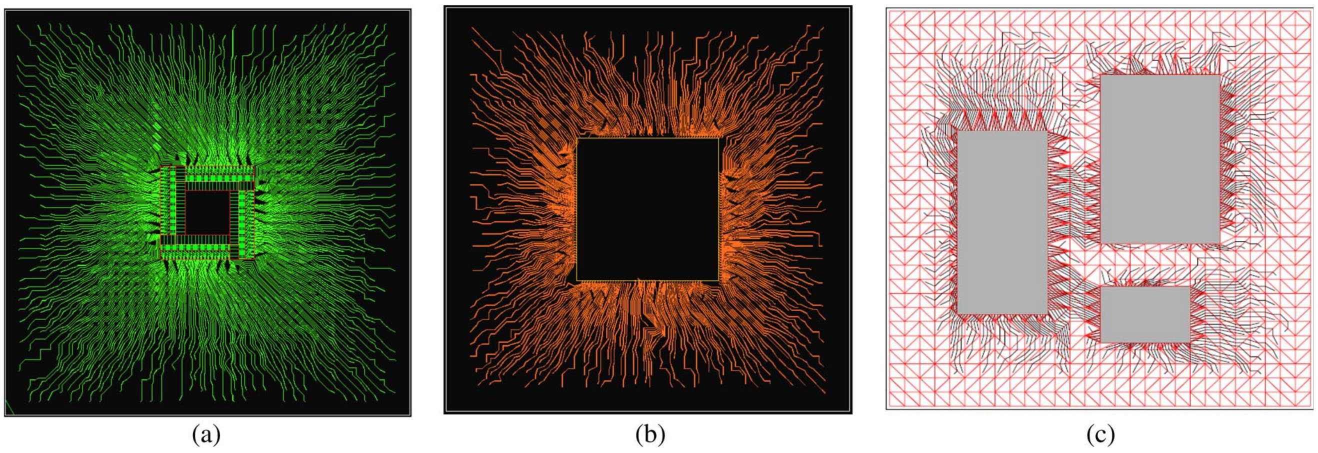


Fig. 15. Topological routing results on a buildup layer. (a) Case 3. (b) Case 4. (c) Case 8.

order A–B–C is used for rerouting, a better solution with shorter length is achieved.

After establishing a high-level order using whole reordering, we still need to perform local changes using *partial reordering*. That is, if we find the case that one net pushes a group of nets in the iteration of all nets routing by ADS^* (i.e., s2.1 in Fig. 5), the last pushed net has the highest priority. For example, in Fig. 14(a), net A pushes group nets B and C. Net C is the last pushed net; we first reroute net C and then net B. However, if we use only whole reordering based on δ, the order is B–C since net B has a larger increase in length because of the push by net A. However, rerouting net C first and then net B is better, and this is an order that can be obtained by partial reordering.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

The proposed algorithm has been implemented in C++ language and integrated into an industrial package design tool set for topological routing. We test our algorithm on industrial package cases. Table I summarizes the package type and size, die size, and total number of nets. The package type indicates the number of buildup and core layers that have the same meaning as that in Section II-A. The first seven test cases are one-die packages, and the remaining two are multiple-die packages. The experiments used a Linux 2.6 server with 2.4-GHz dual central processing units and 2-GB memory.

There is no similar existing work considering the same problem formulation. To illustrate the advantage of our proposed technologies, we compare with the current best known method of topological routing, i.e., BKM, based on the A^* searching algorithm as the best alternative. BKM is substrate routing in an industrial flow, which also has the ripping-up-and-rerouting procedure with A^* routing path searching. We report the number of failed nets, average wire length, and running time in Table I for both algorithms. As shown in the table, our routing algorithm leaves 212 failed nets, whereas BKM leaves 936 failed nets when manually run. Meanwhile, the average wire length of the final routing solution is reduced by 13.9%, on average, by our algorithm. For most test cases, our algorithm has a running time on the same order of magnitude as the BKM algorithm. We also test the routability achieved by dynamic pushing and flexible via staggering, respectively. On average, dynamic pushing reduces roughly by 75% failed nets, and flexible via staggering reduces roughly by 25% failed nets.

In addition to the statistics in Table I, we also plot the topological routing results of cases 3, 4, and 8 on a buildup layer

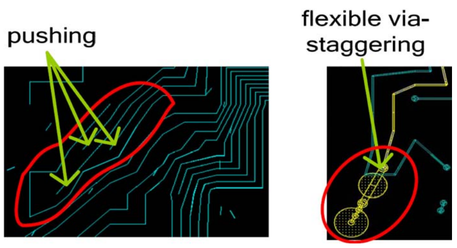


Fig. 16. Partial substrate routing based on dynamic pushing and flexible via staggering.

in Fig. 15(a)–(c), respectively. Case 8 is a complex package. A magnified view of a corner of substrate routing based on dynamic pushing and flexible via staggering is shown in Fig. 16.

We discuss about the advantage of dynamic pushing and rip-upping and routing working together in the algorithm. With dynamic pushing, we can reduce the total wire length or find a solution that the traditional router cannot find, as illustrated in Fig. 8. Fig. 9 and the related explanation in Section III-B1 give a detailed description of how dynamic pushing works. In addition, the routing result considers routability. However, during iterative routing, some of the pushing leads to the problem of bent wires. Then, rerouting with reordering strategies is performed to solve the problem, as presented in Section III-C. Since all the aforementioned operations are based on the evaluation function combining with dynamic programming and the related parameters, the final routing result is with a good tradeoff between pushing and detour. The experimental result in Fig. 16 shows the following: three nets push one net, and it obtains shorter total wire length compared with the result without the pushing technique (i.e., the pushed net directly routes, and the other three nets have to detour).

V. Conclusion

Considering high-density packaging, we have developed a planar topological router. Compared with one current industrial router, our algorithm does not limit start-point locations. We allow routing to finish in a zone or at fixed locations, and honor the solder ball assignment specified for start points. Experiments using an industrial design tool and examples show that the industrial design tool leaves 936 nets unrouted for nine industrial designs with a total of 6100 nets, whereas our algorithm reduces the unrouted nets to 212, a 4.5-times net number reduction, which translates to design time reduction.

Appendix
Proof of Theorem 1

Proof: Inside the end zone, path searching can end anywhere.

Let $g(x)$ be the cost of ending on the arbitrary location x inside the end zone, where x is the distance from the oz to the location.

When path searching decides to end, the estimated cost $ec = 0$, and the total cost is $rc + g(x)$.

The goal of ADS^* is to find the minimum cost path.

Meanwhile, for the sake of staggered via assignment, we assume that the endpoint is preferred to be close to the oz,

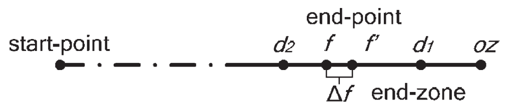


Fig. 17. ADS^* searching with flexible via staggering.

whereas the total cost remains the same.

Thus, ADS^* prefers to end on a location inside the end zone *if and only if moving forward costs more than ending on the current location*.

When moving forward, the actual cost rc increases, and $g(x)$ first decreases (between d_1 and d_2) and then remains 0 (between oz and d_1).

Then, the only motivation of moving forward is that $g(x)$ decreases.

Therefore, rule R1 is obtained, i.e., net path searching always ends between d_1 and d_2.

In the condition shown in Fig. 17, which is depicted in 1-D for simplicity of presentation, let c_0 be the actual cost on the location f.

Then, $c_0 + w \cdot \Delta f$ is the actual cost on the location f' according to (6), where w is the new width when moving forward from location f to f', including the case that some nets will be pushed during moving forward.

Thus, the searching $frontier$ ends if and only if the following condition is satisfied:

$$c_0 + w \cdot \Delta f + g(f') > g(f) + c_0$$

where $g(f)$ and $g(f')$ are the costs caused by ending on the location f and f', respectively.

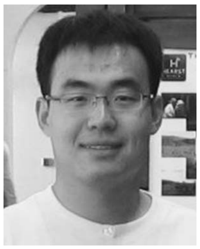
Then, $w > (g(f) - g(f'))/\Delta f$, and $g(x)$ is a linear function.

Hence, $w > ((g(f) - g(f'))/\Delta f) \rightarrow g'(f') = s$, and rule R2 is proved. ■

References

[1] J. W. Fang, I. J. Lin, Y. W. Chang, and J. H. Wang, "A routing algorithm for flip-chip design," in *Proc. Int. Conf. Comput.-Aided Des.*, 2005, pp. 753–758.
[2] J. W. Fang, C. H. Hsu, and Y. W. Chang, "An integer linear programming based routing algorithm for flip-chip design," in *Proc. Des. Autom. Conf.*, 2007, pp. 606–611.
[3] M. M. Ozdal and D. F. Wong, "Simultaneous escape routing and layer assignment for dense PCBs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2002, pp. 822–829.
[4] E. Winkler, "Escape routing from chip scale packages," in *Proc. IEEE Electron. Manuf. Technol. Symp.*, 1996, pp. 393–401.
[5] R. S. Wang, R. Shi, and C. K. Cheng, "Layer minimization of escape routing in area array packaging," in *Proc. Int. Conf. Comput.-Aided Des.*, 2006, pp. 815–819.
[6] M. Horiuchi, E. Yoda, and Y. Takeuchi, "Escape routing design to reduce the number of layers in area array packaging," *IEEE Trans. Adv. Packag.*, vol. 23, no. 4, pp. 686–691, Nov. 2000.
[7] R. Shi and C. K. Cheng, "Efficient escape routing for hexagonal array of high density I/Os," in *Proc. Des. Autom. Conf.*, 2006, pp. 1003–1008.
[8] M. M. Ozdal, M. D. F. Wong, and P. S. Honsinger, "An escape routing framework for dense boards with high-speed design constraints," in *Proc. Int. Conf. Comput.-Aided Des.*, 1995, pp. 759–766.
[9] W. W. Dai, R. Kong, and M. Sato, "Routability of a rubber-band sketch," in *Proc. Des. Autom. Conf.*, 1991, pp. 45–48.
[10] H. F. S. Chen and D. T. Lee, "A faster algorithm for rubber-band equivalent transformation for planar VLSI layouts," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 2, pp. 217–227, Feb. 1996.
[11] D. Staepelaere, "Geometric transformations for a rubber-band sketch," M.S. thesis, Univ. California, Santa Cruz, CA, 1992.

[12] 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 Design Integr. Circuits Syst.*, vol. 17, no. 2, pp. 182–189, Feb. 1998.
[13] Y. Kubo and A. Takahashi, "A global routing method for 2-layer ball grid array packages," in *Proc. Int. Symp. Phys. Des.*, 2005, pp. 36–43.
[14] S. S. Chen, J. J. Chen, S. J. Chen, and C. C. Tsai, "An automatic router for the pin grid array package," in *Proc. Asia South Pacific Des. Autom. Conf.*, 1999, pp. 275–281.
[15] M. F. Yu, J. Darnauer, and W. W. Dai, "Interchangeable pin routing with application to package layout," in *Proc. Int. Conf. Comput.-Aided Des.*, 1996, pp. 668–673.
[16] M. F. Yu and W. W. Dai, "Pin assignment and routing on a single-layer pin grid array," in *Proc. Asia South Pacific Des. Autom. Conf.*, 1995, pp. 203–208.
[17] L. P. Chew, "Constraint Delaunay triangulations," *Algorithmica*, vol. 4, no. 1, pp. 97–108, 1980.
[18] F. Bossen, "Anisotropic mesh generation with particles," M.S. thesis, Univ. CMU, Pittsburgh, PA, 1996.
[19] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
[20] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 346–365, Sep. 1961.
[21] E. F. Moore, "Shortest path through a maze," *Ann. Comput. Lab. Harvard Univ.*, vol. 30, pp. 285–292, 1959.
[22] D. Staepelaere, J. Jue, T. Dayan, and W. W. Dai, "SURF: Rubber-band routing system for multichip modules," *IEEE Des. Test Comput.*, vol. 10, no. 4, pp. 18–26, Dec. 1993.
[23] H. Shin and A. Sangiovanni-Vincentelli, "A detailed router based on incremental routing modifications: Mighty," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-6, no. 6, pp. 942–955, Nov. 1987.



Shenghua Liu (S'08) received the B.S. degree (with the honor of *outstanding graduate* of Shanxi Province) from Xidian University, Xi'an, China, in 2005. He is currently working toward the Ph.D. degree with the Electronic Design Automation Laboratory, Department of Computer Science and Technology, Tsinghua University, Beijing, China, where he completed his B.S. thesis.

He was a Visiting Student with the University of California, Los Angeles, from September 2006 to December 2007. His current research interests include layout algorithm, particularly fast congestion estimation, package routing, RDL routing, and 3-D global routing.

Guoqiang Chen received the B.S. degree in electronics and information system from Peking University, Beijing, China, in 1995 and the M.S. degree in electrical engineering from the University of Minnesota, Minneapolis, in 1998.

He is currently with Magma Design Automation, Inc., San Jose, CA, as a Senior Manager of product development for Magma's chip package codesign tool. He joined Magma as part of the acquisition of Rio Design Automation, Inc., in 2007. At Rio, he was part of the founding engineering team and led the R&D team to bring RioMagic, the industry's first chip package codesign tool, from concept to production. Prior to that, he was with Synopsys, Inc., and Altera, where he worked in various areas of software development. His current interests include VLSI physical design automation and package-aware chip design.

Tom Tong Jing (M'02) received the B.S. degree in electronic engineering and the M.S. and Ph.D. degrees in computer science from the Northwestern Polytechnical University, Xi'an, China, in 1989, 1992, and 1999, respectively.

From 1999 to 2001, he was a Postdoctoral Researcher with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He was a member of the faculty (an Associate Professor from 2001 to 2004 and a Tenured Associate Professor from 2004 to 2006) with the Department of Computer Science and Technology Department, Tsinghua University. He was a Visiting Scholar with the University of California, San Diego, and the Chinese University of Hong Kong, Sha Tin, Hong Kong. From 2006 to 2008, he was a Research Associate with the Department of Electrical Engineering, University of California, Los Angeles. He is currently with Synopsys, Inc., Mountain View, CA. He has authored or coauthored more than 120 papers published in technical journals and conference proceedings. His research interests include computer-aided design for VLSI circuits and systems, combinational optimization and modern optimization methods, and algorithms and data structures.

Dr. Jing served as a Technical Program Committee (TPC) Member of the IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC) 2006; the Secretary General and Chair of the Physical Design and Interconnect Optimization TPC Subcommittee; the Session Chair of the IEEE/ACM ASP-DAC 2005; the Session Chair of the IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP) 2005; a TPC Member, a Panel Speaker, and a Session Cochair of the IEEE International Conference on Communications, Circuits and Systems (ICCCAS) 2004; a Session Chair of the International Symposium Computing and Information (ISCI) 2004; the Secretary General and TPC Member of the IEEE International Conference on ASIC (ASICON) 2003; and a Session Cochair of the IEEE/ACM ASP-DAC 2003. He was a recipient of the IEEE/ACM ASP-DAC Best Paper Award in 2005, the ACM/IEEE International Symposium on Quality Electronic Design (ISQED) Best Paper Nomination in 2005, the IEEE ASICON Outstanding Student Paper Award in 2003, the Second Class Science and Technology Award from the Ministry of Education of China in 2005, the First Class Award for Excellence in Teaching from the Beijing Municipal Education Commission in 2004, and the First Class Awards for Excellence in Teaching from Tsinghua University in 2002 and 2004.

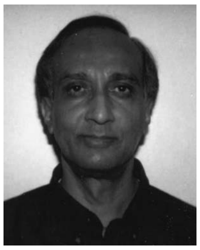
Lei He (M'99–SM'08) received the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA), in 1999.

He is currently an Associate Professor with the Department of Electrical Engineering, UCLA. He was a faculty member with the University of Wisconsin-Madison between 1999 and 2001. He has held visiting or consulting positions with Intel, Hewlett-Package, Cadence, Synopsys, Inc., Rio Design Automation, Inc., and Apache Design Solutions. He has authored one book and more than 170 technical papers. His research interests include VLSI circuits and systems, and electronic design automation.

Dr. He has been a Technical Program Committee Member for a number of conferences, including Design Automation Conference, International Conference on Computer-Aided Design, International Symposium on Low Power Electronics and Design, and International Symposium on Field Programmable Gate Array. He was a recipient of the National Science Foundation CAREER Award in 2000, the UCLA Chancellor's Faculty Career Development Award in 2003, the IBM Faculty Award in 2003, the Northrop Grumman Excellence in Teaching Award in 2005, the Best Paper Award at the 2006 International Symposium on Physical Design, and multiple best paper nominations at the Design Automation Conference and International Conference on Computer-Aided Design.

Tianpei Zhang received the B.S. degree in applied physics from the University of Science and Technology of China, Hefei, China, in 1997, the M.S. degree in electrical engineering from Purdue University, West Lafayette, IN, in 2000, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2006.

He is currently with Cadence Design Systems, San Jose, CA. His research interest includes VLSI physical design, routing, and design for manufacturability.



Robi Dutta received the M.S. degree in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 1971 and the Ph.D. degree in electrical engineering from Concordia University, Montreal, QC, Canada, in 1975, on a fellowship from the National Research Council of Canada.

Before founding Everest Design Automation in 1997, he worked on diverse areas of mathematical modeling and electronic design automation at various companies, including Alcan, Bell-Northern Research, Datapoint Corporation, Digital Equipment Corporation (DEC), and SGI. He has also been closely associated with the academia, including the University of California, Berkeley (UC Berkeley), and the Massachusetts Institute of Technology (MIT), Cambridge. While at DEC, he supervised an M.S. thesis work on physical design at MIT. In addition, during 1986–1987, he was a Visiting Fellow with UC Berkeley. Since the acquisition of Everest by Synopsys, Inc., in 1998, he was a Vice President of Engineering with Synopsys, leading the R&D efforts in various aspects of VLSI physical design until May 2003. In 2003, he cofounded Rio Design Automation, Inc., which was acquired by Magma Design Automation, Inc., San Jose, CA, in 2007. He is currently an independent EDA consultant.

Xian-Long Hong (M'95–SM'95–F'03) received the B.S. degree from Tsinghua University, Beijing, China, in 1964.

Since 1988, he has been a Professor with the Department of Computer Science and Technology, Tsinghua University. He has authored five books and more than 400 papers. His research interests include very large scale integration layout algorithms and design automation systems.

Prof. Hong has served as a Steering Member of the Asia and South Pacific Design Automation Conference (ASPDAC) and a Cochair of the Technical Program Committee of ASPDAC in 1999, 2004, and 2005, respectively. He has been an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I since 2002.