Topological Routing to Maximize Routability for Package Substrate[*]

Shenghua Liu
Tsinghua University
Beijing, 100084, China

Guoqiang Chen
Magma Design Automation, Inc.
San Jose, CA 95110, USA

Tom Tong Jing   Lei He
University of California at Los Angeles
Los Angeles, CA, 90095, USA

Tianpei Zhang
University of Minnesota
Minneapolis, MN, 55455, USA

Robi Dutta
Magma Design Automation, Inc.
San Jose, CA 95110, USA

Xian-Long Hong
Tsinghua University
Beijing, 100084, China

ABSTRACT

Compared with on-chip routers, the existing commercial tools for off-chip routing have a much lower routability and often result in a large number of unrouted nets for manual routing. In this paper, we develop an effective, yet efficient, substrate routing algorithm, applying dynamic pushing to alleviate the net ordering problem and reordering and rerouting for further wire length and congestion reduction. 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 and practically more design time reduction.

Categories and Subject Descriptors

B.7.2 [**Design Aids**]: Placement and routing

General Terms

Algorithms, Experimentation

Keywords

IC package, substrate routing, system in package

1. INTRODUCTION

IC Package usually uses *ball grid array* (BGA) substrate and *wire bonding* or *flip-chip* to connect a microchip to the substrate. However, high-density package integration makes off-chip routing a challenging task. For wire bonding dies, the I/O pads of a microchip are connected to the *bond pads* around the cavity through bonding wires. For flip-chip dies, on-chip *re-distribution layer* (*RDL*) routing [4] first connects the I/O pads to *bump pads*, and *escape routing* then breaks out bump pads to the boundary of the die at the *escape break points* in the build-up or signal layers. Finally, *substrate routing* connects escape break points of flip-chip dies or bond pads of wire bonding dies to *balls* (usually in the bottom layer) of a BGA package substrate.

Substrate routing can be divided into two steps: topological routing, studied in this paper, and detailed routing. We develop an effective, yet efficient, net by net substrate routing algorithm, applying dynamic pushing to alleviate the net ordering problem and reordering and rerouting for further wire length and congestion reduction. Compared with an industrial tool that leaves 936 nets unrouted for nine industrial designs with total 6100 nets, our algorithm reduces the unrouted nets to 212, a 4.5-times net number reduction. The average wirelength reduction by our method is 13.9%.

The rest of this paper is organized as follows. Section 2 formulates the new routing problem. Section 3 describes our algorithms in detail. Section 4 presents experimental results, and Section 5 concludes this paper.

2. PROBLEM FORMULATION

The bond-pads and escape break points are called *start-points* of nets in this paper. Most existing work in topological routing [7, 5, 2] assumed that *start-points* are located side to side with respect to balls. Our routing algorithm to be presented does not suffer from such location constraints.

Existing substrate routing work [9, 8] did not specify the destination ball to a start-point in substrate routing, which makes the routing easier. However, specifying ball assignment is preferred by designers to consider constraints due to printed circuit board (PCB) routing. Our routing algorithm can deal with the case with specified ball assignment.

The substrate routing is preferred to be planar due to signal integrity constraints. The nets are finally connected to the balls in the bottom-layer by staggered vias. The staggered vias usually cannot be stacked exactly one on top of another due to required offset called *staggered via pitch* that has the constraint of minimum and maximum staggered via pitches. The former is determined by the via manufacturing technology. The latter is determined by the P/G network since the pitch should not impact on the integrity of P/G plane. Thus, taking a typical four-two-four package (i.e., in a ten-layer package, from top to bottom, there are four build-up layers, two core layers, and four build-up layers. It can be used as GND-Signal-Vdd-Signal-GND-Vdd-Signal-GND-

[*]This paper is partially supported by NSF CAREER award CCR-0093273. The work by Mr. Liu and Mr. Zhang was performed during their internships with Rio Design Automation, Inc., now a part of Magma Design Automation, Inc., and more details and updates of this work are available at http://eda.ee.ucla.edu. Address comments to *gchen@Magma-DA.COM* and *tomjing@ucla.edu.*

Signal-Vdd.) for example, we have the flexibility to decide where the planar routing ends and staggered via starts. We define the zone where staggered via can be located in the signal layer as the *end-zone*. The *end-zone* includes two circles (see Figure 1). 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 pitch in the layer with index i, respectively. However, most existing work [7, 2] ignored the need to stagger vias by simply routing to a fixed end point. In [5], via assignment was performed ahead of routing, but it is suitable only for the case of fixed via location with two-layer packages. Our work performs via staggering so that a net can be connected to any point in its end-zone defined by its ball assignment, and improves routability compared with the existing work connecting the net to a fix point inside the end-zone.

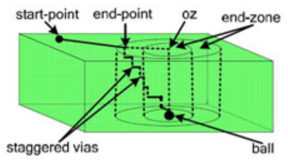

Figure 1: End-zone and the flexible end-point.

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

FORMULATION (*Substrate Topological Routing*) *Given start-points, balls in the bottom layer, netlist defined by ball assignment, and obstacles (including the escape area for escape routing, pre-routed 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 ball, such that the routed nets inside the presence of obstacles are planar, satisfying the capacity constraints, and have minimal wire length.*

In our routing algorithm, the SRG in a build-up 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, pre-routed connections, and all kinds of possible polygons on the SRG plane. Considering these practical constraints, we apply a triangle mesh by constraint Delaunay triangulation (CDT) [3] in this paper, which guarantees a low computational cost and reasonably well-shaped elements. Uniformly spreading points, called *particles* U, are added in the same way as [1] 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), obstacles, particles U, and the boundary of the SRG plane. Thus, the start-points and the *oz*s become vertices of triangles. The dual graph of PCDT is built accordingly, which we name D-PCDT. An example of PCDT and its dual graph D-PCDT are shown in Figure 2.

3. ALGORITHMS

The overview of our topological routing algorithm is shown in Figure 3. There are two core algorithms: (1) dynamic searching algorithm DS^* (called *DS star*) and (2) reordering algorithm. The two algorithms are carried out layer by layer from top to bottom. Initial routing and rerouting both employ the DS^* algorithm. *Bound of pushes for a single net*, ξ, is given as a threshold in each iteration to control pushing. ξ starts at the infinity, as s1.5 in Figure 3, and is gradually reduced to 0 as in s2.2.2. To reduce congestion, we use ψ as a *congestion threshold* to guide DS*. Highly congested edges with higher *congestion value* η_{ed} (discussed in Section 3.1) are forbidden from being passed. 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.

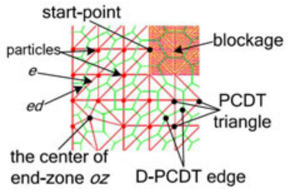

Figure 2: Partial PCDT and D-PCDT.

```
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 $u$ 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 dynamic search $DS^*(\psi, \xi)$ for the net path;
    }
    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;
  }
}
```

Figure 3: Algorithm overview.

3.1 Dynamic Searching Algorithm DS*

To route net by net, we develop a searching algorithm, called dynamic searching algorithm DS*. For each two-pin net, DS* finds a shortest path on the D-PCDT graph (on the PCDT graph, it is a path from one triangle to another one) 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, or on the boundary of the obstacle, or on the boundary of SRG plane, then, $C_{ed} = 0$. Otherwise, $C_{ed} = l_e$, where l_e is the length of the edge e. The congestion of edge ed is defined as follows.

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

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

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

DS* is a local search algorithm for net routing which makes use of dynamic programming technique with an *evaluation function* (i.e., the sum of actual cost and estimated cost). The evaluation function is designed based on two key technologies: *dynamic pushing* and *flexible via-staggering*. Then, the evaluation function can guide the search to "*push*" or "*detour*" the routed nets blocking the current net that is searching its path. We use a heap in the implementation of DS*. Once the path on D-PCDT graph is found, cross points on PCDT graph are assigned simultaneously.

3.1.1 Searching with dynamic pushing

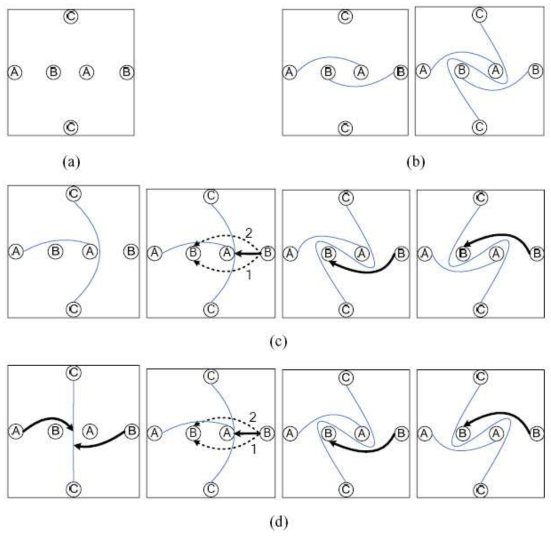

Figure 4: An example of net ordering (a) nets initial location, (b) net order (A, B)-C and the routing, (c) net order A-C-B, and the routing, (d) net order C-(A, B) and the routing. Only this work can obtain routing (c) and (d).

To tackle the net ordering problem, we embed a "*dynamic pushing*" technique during the net path search. We use Figure 4 to illustrate how dynamic pushing works. Net A and net B in Figure 4(a) is 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 be solved directly by a traditional router such as Rubber-Band method [6] shown in Figure 4(b). However, dynamic pushing can handle net orders that the existing work cannot consider. For the net order A-C-B, after DS* algorithm connects net A and net C, net B "*pushes*" routed net C and uses either path 1 or path 2 shown in Figure 4(c). For the net order C-(A,B), after DS* 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 Figure 4(d).

3.1.2 Searching with flexible via-staggering

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

As Figure 5(a) shows, in the traditional fixed end-point case, net B must detour around routed net A to finish connection. But the flexible via-staggering technique of DS* can both successfully route net B and obtain shorter wire length by changing the end-point of net A when the end-point is flexible as shown in Figure 5(b). Figure 5(c) gives the 3D description of flexible via-staggering technique. Figure 5(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 end-point of net A shown in Figure 5(e). Therefore, the flexible via-staggering can obtain higher routability and shorter wire length. Moreover, it has no extra expense since it makes use of the required via-staggering pitch.

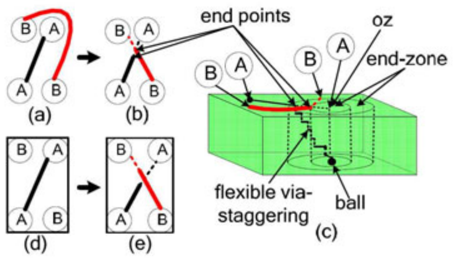

Figure 5: Comparison between fixed end-point ((a) Detour and (d) Unroutable) and flexible via-staggering ((b) Non-detour, (c) 3D description, and (e) Routable).

3.2 Reordering Strategy

Based on our experiments, we observe that routing nets frequently results in *bent wires* caused by pushing, as demonstrated in Figure 6(a) and Figure 7(a). *Bent wires* usually involve unnecessary detours and increase total wire length. We solve this problem by using rip-up and rerouting based on reordering. That is, after one iteration of all nets routing by DS* (see s2.1 in Figure 3), we reorder all nets (see s2.3 in Figure 3). Then we rip-up one net, leaving others still routed, and reroute it by DS*(ψ, ξ) as s2.1.1 and s2.1.2 in Figure 3 show.

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

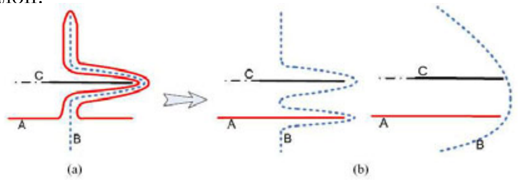

Figure 6: Reordering and finding a better solution. (a) net A is pushed twice, (b) a better solution.

We define a *net length ratio* $\delta = l/u$, where l is the net

Table 1: TEST CASES AND EXPERIMENTAL RESULTS

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

Name of circuit	Package type	Package size*	Die(s) size*	Number of nets	# of failed nets		Runtime(s)		Average WL (μm)	
					BKM	new	BKM	new	BKM	new
case 1	2-2-2	35000 × 35000	14000 × 15000	474	52	16	2.06	11.33	9.10E+03	8.49E+03
case 2	2-2-2	30000 × 30000	9000 × 10500	543	10	0	1.90	5.34	8.24E+03	7.99E+03
case 3	3-1-3	40000 × 40000	9300 × 9300	800	306	49	13.81	16.86	1.65E+03	1.37E+03
case 4	3-2-3	35000 × 35000	12000 × 12000	506	82	6	6.90	8.46	2.29E+04	1.78E+04
case 5	3-2-3	40000 × 40000	20000 × 22000	891	98	45	2.60	13.65	5.93E+03	5.27E+03
case 6	4-2-4	40000 × 40000	20000 × 23000	990	198	51	9.95	26.62	9.10E+03	7.93E+03
case 7	4-2-4	45000 × 45000	20000 × 19000	1009	100	14	5.91	26.14	2.31E+04	1.89E+04
case 8	1-0-1	12000 × 12000	3900 × 6700 4400 × 5700 3200 × 4400	349	60	22	15.24	1.51	1.83E+03	1.73E+03
case 9	2-2-2	37500 × 37500	11000 × 10000 4700 × 3800 4600 × 5500	538	30	9	95.17	110.93	9.78E+03	9.38E+03
total	—	—	—	6100	936	212	—	—	—	—
average	—	—	—	—	—	—	—	—	1.02E+04	8.76E+03 (-13.9%)

length acquired from the latest routing iteration. We first use *whole reordering*, i.e., *largest δ first*, to generate a new net order, which prioritizes nets with larger increases caused by *pushing*. For example, in Figure 6(a), net A is pushed first by net B and then by net C. Thus, nets such as net A are allocated a higher priority to allow them to "*stretch*". In Figure 6(b), if using the order of A-B-C for rerouting, a better solution with shorter length is achieved.

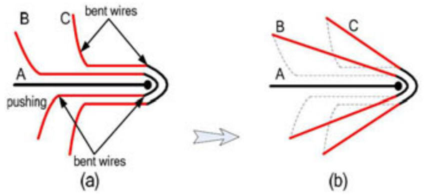

Figure 7: (a) Bent, (b) stretch.

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 *DS** (i.e., s2.1 in Figure 3), the last pushed net has the highest priority. For example, in Figure 7(a), net A pushes group nets B and C. Net C is the last pushed net; we reroute net C first and then net B. But 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*.

4. EXPERIMENTAL RESULTS

The proposed algorithm has been implemented in the C++ language and integrated into an industrial package design tool set for topological routing. We test our algorithm on industrial package cases. Columns 2-5 of Table 1 summarize the package type and size, die size, and total number of nets. The package type indicates the number of build-up layers and core layers that have the same meaning as that in Section 2. The first seven test cases are one-die packages and the remaining two are complex packages. The experiments used a Linux-2.6 server with 2.4GHz dual CPUs and 2GB 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, BKM, based on the $A*$ searching algorithm and ripping-up and rerouting as the best alternative. We report the number of failed nets, running time, and average wire length in Table 1 for both algorithms. As seen in the table, our routing algorithm leaves 212 failed nets, while BKM leaves 936 failed nets when run manually. 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 in 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 75% failed nets and flexible via-staggering reduces roughly 25% failed nets, respectively.

5. CONCLUSIONS

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 the routing to finish in a zone or at fixed locations, and honor the 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, while our algorithm reduces the unrouted nets to 212, a 4.5-times net number reduction and practically more design time reduction.

6. REFERENCES

[1] F. Bossen. Anisotropic mesh generation with particles. In *M.S. thesis, Univ. of CMU*, 1996.

[2] 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 Design Automation Conf.*, pages 275–281, 1999.

[3] L. P. Chew. Constraint Delaunay triangulations. *Algorithmica*, 4(1):97–108, 1980.

[4] J. W. Fang, C. H. Hsu, and Y. W. Chang. An integer linear programming based routing algorithm for flip-chip design. In *Proc. Design Automation Conf.*, pages 606–611, 2007.

[5] Y. Kubo and A. Takahashi. A global routing method for 2-layer ball grid array packages. In *Proc. Intl. Symp. Physical Design*, pages 36–43, 2005.

[6] D. Staepelaere, J. Jue, T. Dayan, and W. W. Dai. SURF: a rubber-band routing system for multichip modules. *IEEE Trans. Design & Test of Computers.*, 10(4):18–26, 1993.

[7] 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. Computer-Aided Design of Integrated Circuits and Systems*, 17(2):182–189, 1998.

[8] M. F. Yu and W. W. Dai. Pin assignment and routing on a single-layer pin grid array. In *Proc. Asia South Pacific Design Automation Conf.*, pages 203–208, 1995.

[9] M. F. Yu, J. Darnauer, and W. W. Dai. Interchangeable pin routing with application to package layout. In *Proc. Intl. Conf. Computer-Aided Design*, pages 668–673, 1996.