Diffusion-Driven Congestion Reduction for Substrate Topological Routing*

Shenghua Liu
Tsinghua University
Beijing, 100084 China

Guoqiang Chen
Magma Design Automation
San Jose, CA 95110 USA

Tom Tong Jing Lei He
UCLA
Los Angeles, CA 90095 USA

Robi Dutta
Magma Design Automation
San Jose, CA 95110 USA

Xian-Long Hong
Tsinghua University
Beijing, 100084 China

ABSTRACT

Off-chip substrate routing for high density packages is challenging, and the existing substrate routing algorithms often result in a large number of unrouted nets that have to be routed manually. This paper develops an effective yet efficient diffusion-driven method D-Router to improve routability by a simulated diffusion process based on the duality between congestion and concentration. Compared with a recently published A*-based algorithm used in a state of the art commercial tool and with similar routability and runtime as the negotiation based routing, D-Router reduces the number of unrouted nets by 4.6x with up to 94x runtime reduction.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and Routing*

General Terms

Algorithms, Design.

Keywords

IC package, substrate routing, routability, congestion reduction, diffusion.

*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. The work by Mr. Liu was performed during his internship with Rio Design Automation, Inc., now a part of Magma Design Automation, Inc. Address comments to gchen@magma-da.com and tomjing@ucla.edu.

ISPD'09, March 29–April 1, 2009, San Diego, California, USA.

1. INTRODUCTION

Package design, an essential part of chip implementation, includes IO placement [13], escape routing [10] for flip-chip dies, and substrate routing [2, 6, 12, 14, 15] for both wire-bonding and flip-chip dies. This paper studies substrate routing, which connects two-pin nets between escape break points of flip-chip dies or bond pads of wire bonding dies to balls in the bottom layer of a package. Since vias could be harmful to signal integrity in high speed signaling and vias have larger widths than wires, vertical detour is not allowed and substrate routing is primarily planar although multiple routing layers are available in the package. To accumulate high density planar routing, non-Manhattan routing becomes a necessity.

Planar and non-Manhattan routing, in addition to technology scaling induced high-density, makes substrate routing a challenging task. Compared with on-chip routers, the existing substrate routing algorithms have a much lower routability and often result in a large number of unrouted nets for manual routing. For example, a very recent substrate routing algorithm displayed a good reported routability in the literature and is used in a state of the art commercial tool. It proposed "dynamic pushing" to tackle the routing order problem and "flexible via staggering" to improve the routability, resulting in 3.5% net unrouted for nine industrial designs [7]. However, the congestion reduction method of iteratively avoiding routing through congested area in [7] limited its advantage in routability.

This paper develops a diffusion based routing (D-Router). Starting with an initial routing without considering any congestion constraint, we iteratively find a congested window and spread out connections to reduce congestion inside the window by a simulated diffusion process based on the duality between congestion and concentration. The window is released after the congestion is eliminated. Compared with the recent substrate routing [7], D-Router reduces the number of unrouted nets by 4.6x, with up to 94x runtime reduction.

The earlier substrate routing Surf system [11] applies topological routing to generate a rubber-band sketch [5], and transforms the sketch first to a spoke sketch and then to a precise geometrical layout. Surf assumes a fixed end point, while our formulation in this paper and [7] uses end-zone, though geometrical layout is not considered in both papers.

Our topological routing formulation is more flexible and therefore increases routability. In addition, Surf completes topological routing with a global routing stage followed by a local routing, while our D-router uses iterative congestion reduction by diffusion without partitioning and avoids the problem of fixing congestion *only* within each bin during local routing in Surf.

To better appreciate D-Router, we also implement a negotiation based substrate routing, which obtains similar routability and runtime as [7] and therefore is inferior to D-Router. In addition to negotiation based routing [8, 9], a recent on-chip router, BoxRouter, also achieves good routability [4], where all nets within a congested window are ripped-up and rerouted simultaneously by an integrity linear programming (ILP) method. However, the ILP method assumes Manhattan routing, and extension to non-Manhattan substrate routing is unclear. While both reduce congestion within a window, D-Router and BoxRouter have the following differences. D-Router essentially rips-up and reroutes wire segments net-by-net, and not necessarily reroutes all nets inside a window. BoxRouter, on the other hand, rips-up and reroutes all nets simultaneously. BoxRouter also expands the window, while D-Router iterates window by window.

The rest of this paper is organized as follows. Section 2 formulates the problem and introduces baseline algorithms for comparison. Section 3 introduces the diffusion method of D-Router in detail. Section 4 presents experimental results, and Section 5 concludes this paper.

2. PRELIMINARIES AND PROBLEM FORMULATION

We use the same problem formulation and data structure as [7]. These as well as baseline algorithms for comparison are discussed in this section.

2.1 Problem Formulation

We first define terms needed for problem formulation.

start-point: The escape break point or bond pad of a net is its *start-point*. In substrate routing, there is little freedom to do layer assignment because the escape routing or wire bonding has already decided the the start-point layer of a net.

end-zone: Instead of connecting a net from its start-point to a fixed end-point, often the location above the center of the assigned ball but in the same layer as the start-point, staggered vias allow connection to any point within a a circle called end-zone. Its center is the aforementioned end-point. Its radius is based on layout rules for staggered vias as discussed in [7].

netlist: connections definition between start-points and end-zones.

obstacles: For each build-up or signal layer, obstacles include the escape area for escape routing, wire-bonding area for bonding wires, pre-routed connections and vias, and other area that cannot be used by routing.

Assuming that the layer assignment is given by the escape routing or wire bonding, the problem of substrate routing is equivalent to a single layer routing, which is performed on the planar substrate routing graph (SRG). For each layer, SRG maps start-points, end-zones, and obstacles on a graph within the rectangular boundary of package substrate. The problem of substrate topological routing is formulated as follows.

FORMULATION 1. *(Substrate Topological Routing) Given start-points, end-zones, and obstacles on a SRG graph, and netlist, find a topological routing solution connecting each start-point to any point in the end-zone defined by netlist, such that the routed nets satisfy the capacity constraints, and have minimal wire length, no vertical detour and no intersections.*

2.2 Data Structure

The SRG graph for each signal layer is further discretized by a set of simple elements, triangles, in two-dimensions. We apply a triangle mesh using the constraint Delaunay triangulation (CDT) [1] 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 [7] to the SRG plane for particle-insertion-based CDT (PCDT) construction. Represented in PCDT, a net path passes through common edges from one triangle to another, and crossing points are assigned on the triangle's edges. Thus, a triangle edge constrains the number of nets passing through it. Let the capacity of edge e of PCDT be C_e. If edge e is on the boundary of an obstacle, or on the boundary of the SRG plane, $C_e = 0$. Otherwise, $C_e = l_e$, where l_e is the length of edge e. Then, the *congestion value* η_e of edge e is defined.

If C_e equals 0, edge e cannot have any net passing through it and end-point located on. We define $\eta_e = 0$, so that those edges are not considered reducing congestion. Otherwise,

$$\eta_e = \frac{\sum_i (w_i + s_i)}{c_e} \tag{1}$$

where w_i and s_i are the wire segment/end-point(*i.e.* via) width and space of net i that passes through edge e, respectively.

2.3 Baseline Algorithms for Comparison

We compare to [7], a very recently published substrate topological routing used in a commercial tool. It routes net by net based on A* algorithm with dynamic pushing and flexible via-staggering. It also applies post-routing rip-up-and-reroute iteration for congestion reduction.

We also compare with negotiated-based routing, where all nets are ripped up and re-routed iteratively according to a fixed ordering, and each net is re-routed with cost function considering the current congestion and congestion history. This routing algorithm has obtained high-quality solutions to on-chip routing of FPGA [8] and ASIC [3, 9]. For comparison in this paper, we use the same substrate routing formulation from [7] and enhance the A*-based algorithm from [7] to re-route a net that always has two pins in substrate routing. The original cost for a node in the routing graph is

$$NC_e = rc + ec \tag{2}$$

where rc and ec are the realized and estimated costs defined in [7]. To consider congestion history required by negotiation based routing, the new cost is defined as

$$NC'_e = (rc + h_e) \times p_e + ec \tag{3}$$

where p_e reflects the present congestion, and h_e represents the congestion history. h_e is given by

$$h_e^{k+1} = \begin{cases} h_e^k + h_{inc}, & \text{if } e \text{ has overflow} \\ h_e^k, & \text{otherwise} \end{cases} \quad (4)$$

with h_{inc} as a constant.

3. DIFFUSION

The planar part of substrate routing is completed by the following algorithm. And staggered vias are used for the vertical connections to the bottom balls.

3.1 Diffusion process

Diffusion happens inside an isolated material. The physical process of dopant diffusion is driven by the concentration gradient, and is defined by the slope and steepness of the concentration difference at a given point. It reaches equilibrium when the material concentration is evenly distributed. The commonly used model of diffusion is *continuous*, *global*, and *simultaneous*. Our algorithm however, is discrete, localized, and works net-by-net. In order to simulate the process of dopant diffusion for congestion reduction in the routing phase, given moment t, we define the *concentration* $d_e(t)$ of PCDT edge e as congestion on edge e for moment t.

$$d_e(t) = \eta_e(t) \quad (5)$$

Below, we will define key concepts such as diffusion window and show that our diffusion is discrete and localized.

3.1.1 Diffusion window

Congestion caused by net segments and net end-points is discrete according to values w_i and s_i in Equation (1). We use the concept of *diffusion window* as an isolated area for congestion reduction. Given a highly congested PCDT edge e, the diffusion window includes edge e itself and adjacent edge sets E_1 and E_2. Edges in E_1 and E_2 are incident to vertices v_1 and v_2 of edge e, respectively. The diffusion window is shown with solid lines in Figure 1. Let edge e be the *diffusion source*, always the most highly congested edge in a validated diffusion window.

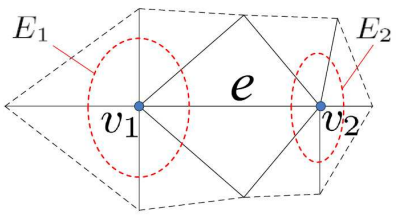

Figure 1: A diffusion window for edge e.

Once a *diffusion window* is created, it is isolated from the outside: *i.e.*, there is no *congestion diffusing* between the outside and inside of the window until the window is released. In the diffusion window, the diffusion is in a one-dimensional (1-D) space with two directions towards E_1 and E_2. In order to guarantee edge e is the diffusion source in *diffusion window*, the most highly congested edge (i.e., with the largest $\eta_e(t)$) is chosen in every iteration.

3.1.2 Diffusion concentration inside window

To solve the discrete routing problem, we discretize the diffusion process within a diffusion window into finite atomic movements of a net segment or a net end-point towards E_1 or E_2 (i.e., away from the diffusion source e). We then develop a practical concentration formula based on Equation (5). When a net moves towards set E_1 or E_2, it may pass through more than one edge in set E_1 or E_2. To decide which direction to diffuse, we first define the *diffused edges* as the edges in E_1 or E_2 through which the net passes. We then consider either side of the diffused edges as a grouped edge and denoted as Edf. Then, we calculate the concentration value of Edf, $d_{Edf}(t)$, by using the maximal congestion of the edge among diffused edges.

$$d_{Edf}(t) = \max_{e_i \in Edf} \{\eta_{ei}(t)\} \quad (6)$$

3.1.3 Diffusion velocity

The congestion reduction is formulated as a localized congestion diffusion problem. For position x, a general definition of 1-D *diffusion velocity* $v_x(t)$ is as follows.

$$v_x(t) = -\frac{d(d_e(t))}{dx}/d_e(t) \quad (7)$$

Then, from Equations (6) and (7), we have the discrete form of velocity for a diffusion source e.

$$v_{e^+}(t) = -(d_{Edf^+}(t) - d_e(t))/d_e(t) \quad (8)$$

$$v_{e^-}(t) = -(d_{Edf^-}(t) - d_e(t))/d_e(t) \quad (9)$$

where Edf^+ and Edf^- are the diffused edges in E_1 and E_2, respectively, and we assume that the moving distance of the atomic movement is 1.

Equations (8) and (9) indicate that the diffusion towards the diffused edges with less concentration has a higher speed. Therefore, in the discrete case, we always select the direction with higher speed to perform the diffusion, which means lower concentration and lower congestion.

3.2 Algorithm overview

The algorithm overview of D-Router is given in Figure 2, and key algorithm components are discussed in Subsections 3.3 - 3.5. In addition to the procedure of local diffusion in the diffusion window, a heap H and a taboo list Tb are used for the convergence of a local diffusion iteration, which are discussed in Section 3.5. A variable congestion threshold ψ is used to solve the congestion accumulating problem, discussed in Section 3.5. ψ decreases from ψ^* until there is no congested area left. A lower bound of the threshold ψ_0 (can be 0) is used to terminate the iteration while meeting irremovable congestion. Decreasing step $\Delta\psi$ can be assigned as $\psi^*/25$.

3.3 Momentary-diffusion operations

The *momentary-diffusion* is the atomic net segment/end-point movement from a diffusion source to selected diffusion direction in the routing problem. We define the momentary-diffusion operations as $D(n, e, E)$, where n is the net segment to be moved, e is the diffusion source, and E is the diffusion direction such as E_1 or E_2 in Figure 1 that we choose at the moment. Let vertex v such as v_1 or v_2 in Figure 1 be the common vertex incident to edge e and edges set E_1 or E_2. Net segment n should be the one closest to

```
INPUT
  Initial routing solution,
  PCDT construction on SRG graph,
  The given congestion threshold ψ*,
  A lower bound of congestion threshold ψ0,
  The congestion threshold decreasing step Δψ
FOR the threshold ψ decreases from ψ* to ψ0 by step Δψ,
DO
{
  IF(no congested edges determined by the threshold ψ*) Break;
  Create a heap H and a taboo list Tb;
  Insert congested edges determined by the threshold ψ into H;
  FOR the most highly congested edge e from H,
  DO
  {
    Find the diffusion window of e;
    WHILE( !(diffusion window reaches equilibrium) )
    {
      Choose the diffusion direction with the higher speed
            at the moment t;
      DO momentary-diffusion D(n, e, E);
      Update the congestion value η_ei(t), e_i ∈ Edf;
      Update the taboo list Tb and the heap H;
    }
    Release the diffusion window;
    IF(e is still congested) put e into Tb;
    Delete e from H;
    Re-heapify H;
  }
}
```

Figure 2: The algorithm of D-Router.

vertex v. Below, we describe the momentary-diffusion operations in each case. For simplicity of presentation, we take direction E_1 and vertex v_1 as an example.

Case 1: Normal

In this case, net n moves through diffusion source e, no net starts from or stops at vertex v_1, and no edge in E_1 is the boundary of a blockage.

The *cycle-canceling* method from network flow optimization can be used to maintain a feasible routing solution at every iteration. A net in PCDT can be regarded as a flow from start point to end-zone. Therefore, the cycle-cancelling method is used to change the segment path. As shown in Figure 3(a), a counter-clock-wise cycle is found with respect to the direction and location of the net flow. Afterward, crossing points are added to the closest positions to vertex v_1 in order to keep the original topology of all nets. Then, crossing points are stitched sequentially to form a chain as shown in Figure 3(b). Finally, a new routing path of net n is found and the congestion of edge e is reduced.

Case 2: Net n stops at a diffusion source e

In this case, net n stops at diffusion source e with a via as shown in Figure 3(c). According to our data structure, when a net stops on a triangle edge, this net must have a flexible end-point and arrive at the end-zone. After adding crossing points on diffused edges in the same way as Case 1, we search those crossing points sequentially until we find a crossing point located inside the end-zone. The searched crossing points are stitched back to the net, forming a new path. Thus, after deleting the redundant crossing points, a new routing result with reduced congestion for net n is obtained in Figure 3(d).

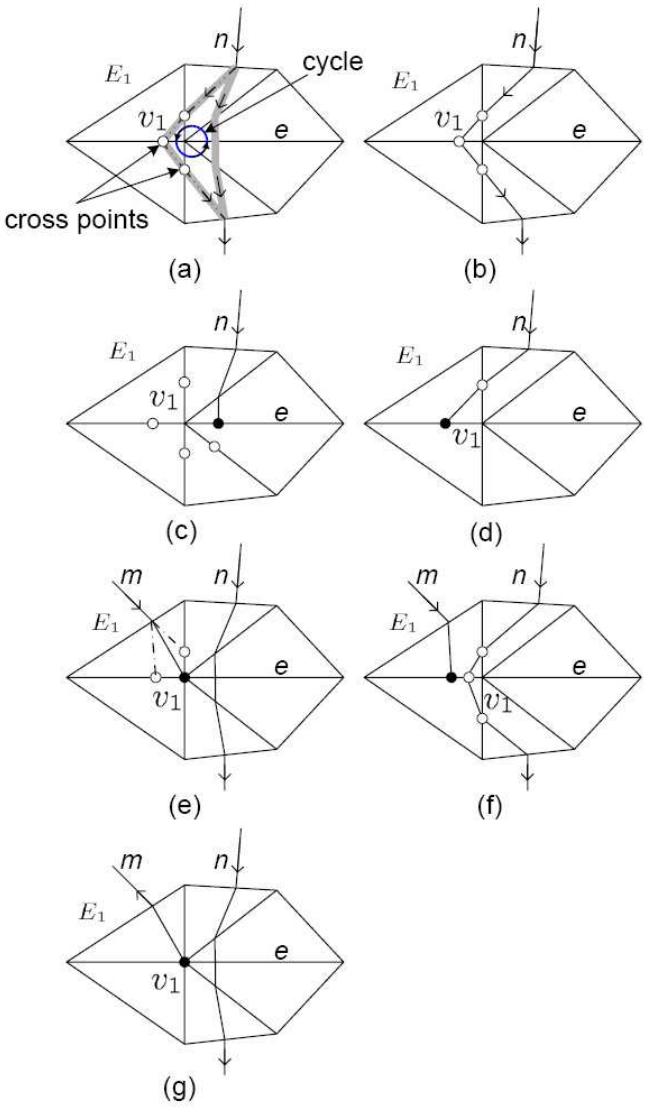

Figure 3: Momentary-diffusion operations. (a) (*cycle-cancelling*) and (b) for *Case* 1, (c) and (d) for *Case* 2, (e) and (f) for *Case* 3, (g) for *Case* 4.

Case 3: Net m stops at vertex v_1 beside net n

According to the data structure PCDT, flexible end-points can be located at vertices as shown in Figure 3(e). In this case, the end-point of net m is released to the less congested edge. This release does not change the topology of net m and a new solution of net n with reduced congestion can be easily obtained by the cycle-cancelling method in Case 1, as illustrated in Figure 3(f).

Case 4: Net m starts at vertex v_1 beside net n

According to data structure PCDT, a start-point must be located at a triangle vertex. Thus, if vertex v_1 is the start-point of net m, net n cannot move to set E_1 as shown in Figure 3(g). In this case, we should try the other side E_2 for the diffusion. If the same case appears in the other side, it is usually the result of improper ball assignment or im-

proper start-point locations decided by escape routing. Such information can be fed back to the escape routing stage.

3.4 Diffusion equilibrium

Any of the following three conditions indicates equilibrium.

Condition I: For a given threshold ψ^*, if congestion $\eta_e(t) < \psi^*$, diffusion reaches equilibrium.

Condition II: The congestion values of every edge in a diffusion window are not necessarily equal to each other when equilibrium is reached. *Over diffusion* is a momentary-diffusion that makes the diffusion source less congested than an edge in set *Edf*. Localized diffusion reaches equilibrium when the next momentary-diffusion is over diffused. Therefore, diffusion repeated between two edges back and forth can be avoided.

Condition III: When diffusion source edge e has one vertex v that is the start-point of a net, the diffusion towards this direction cannot be performed. Additionally, when any diffused edge in *Edf* is the boundary of a blockage, or is forbidden by the taboo list (see Figure 2 and detailed discussion in next subsection), the diffusion towards this direction cannot be made. When diffusion cannot be made in either direction, diffusion reaches equilibrium.

3.5 Iteration convergence

A heap H and a taboo list Tb are maintained for the process of diffusion. H maintains all possible diffusion sources and is heapified by edge congestion. Once the heap H is built, the first element inside is the most highly congested edge. Taboo list Tb maintains all the edges that are no longer allowed to diffuse congestion. When one of the diffused edges *Edf* is such an edge, momentary-diffusion towards this direction is forbidden. In the beginning, H contains all congested edges as possible diffusion sources and Tb is initially empty. When the localized diffusion reaches equilibrium, a diffusion source is added into Tb if still congested. An edge in Tb can be released when the concentration of one of its neighbor edges becomes reduced. Every diffusion source is removed from H once its diffusion window reaches equilibrium. Meanwhile, both newly congested and newly released edges are added into H after every momentary-diffusion. Once H is empty, the planar diffusion process terminates.

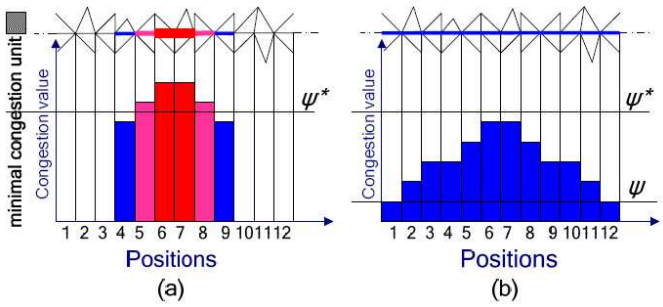

Figure 4: Congestion accumulation effect and improvement.

Our diffusion method is based on discrete concentration and a localized diffusion window. Therefore, if several neighboring diffusion windows reach equilibrium in *Condition* II, the congestion on those neighboring edges accumulate, as in positions 4-6 in Figure 4(a). Meanwhile, edges, as in position 4 in Figure 4(a) with congestion less than ψ^*, are not diffused according to *Condition* I. Thus, congestion easily accumulates more than ψ^* but cannot be reduced, shown in positions 5-8 in Figure 4(a). To solve this, we give a lower congestion threshold ψ as shown in Figure 4(b). Since ψ is lower, the diffusion in the diffusion window of positions 4 and 9 should be continued (*Condition* I is unsatisfied). Also, some net segments in positions 4 and 9 are moved to position 3 and 10, resulting in a larger concentration gradient between positions 4-9. This facilitates diffusions in other windows. Then, congestion can be reduced further, and the final result through iteratively diffusion is shown in Figure 4(b), which spreads routed nets more evenly.

4. EXPERIMENTAL RESULTS

Table 1: TEST CASE CHARACTERS
(*: Package size and Die(s) size are given by width × length (μm) in rectangle.)

Test case ID	Package type	Package size*	Die(s) size*	Number of nets
Q1	2-0-2	10000×10000	75007×700	315
B2	2-2-2	35000×35000	14000×15000	474
F3	2-2-2	30000×30000	9000×10500	543
P4	3-1-3	40000×40000	9300×9300	800
A5	3-2-3	35000×35000	12000×12000	506
A6	3-2-3	40000×40000	20000×22000	891
X7	4-2-4	40000×40000	20000×23000	990
A8	4-2-4	45000×45000	20000×19000	1009
S9	1-0-1	12000×12000	3900×6700 4400×5700 3200×4400	349
S10	2-2-2	37500×37500	11000×10000 4700×3800 4600×5500	538
total	—	—	—	6415

Table 1 summarizes the test case characteristics, including 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. The package substrate always keeps a symmetric structure, for the balance of thermal dilation. Thus, a type of m-n-m substrate indicates that there are n core layers and $2m$ build-up layers. All the core layers and some of build-up layers are used for P/G planes, while the other build-up layers are used for signal routing, *i.e.* substrate routing. The first eight test cases are single-chip packages while the remaining two have multiple dies within the packages. The last nine test cases are from [7] which similar to D-Router, has already given quite a good solution. However, the test cases that we use to compare are different from those in [7]. The experiments use a Linux-2.6 server with 2.4GHz dual CPUs and 2GB memory.

Table 2 compares D-Router with [7] and the negotiation-based substrate routing (called "Nego" in Table 2) introduced in Section II.C. We do not compare to Surf since problem formulations are different and the effort to re-implement Surf for our problem formulation is hardly justified. D-Router has 104 failed nets while the other two algorithms have failed 480 and 461 nets, respectively. We also measure the wire length calculated for all nets that can be routed by the three algorithms, and they obtain similar length. In

Table 2: EXPERIMENTAL RESULTS
(Nego: negotiation-based substrate routing)

Test case	Number of failed nets			Wire length (mm)			Runtime (s)		
	[7]	Nego	D-Router	[7]	Nego	D-Router	[7]	Nego	D-Router
Q1	51	41	26	1.64	1.69	1.70	5.34	9.79	7.17
B2	31	30	0	6.98	6.98	7.17	11.39	17.84	9.00
F3	24	22	0	7.79	7.80	7.96	14.36	14.41	16.91
P4	135	135	48	11.90	11.90	12.30	41.64	20.04	13.87
A5	64	63	7	14.90	14.90	16.30	15.27	17.65	10.92
A6	60	57	15	4.98	4.99	4.93	12.12	18.77	12.87
X7	45	45	8	6.55	6.54	6.53	39.51	45.11	25.38
A8	16	16	0	18.50	18.50	18.70	44.55	47.24	9.32
S9	22	20	0	1.67	1.67	1.65	2.11	3.2	0.96
S10	32	32	0	9.53	9.53	7.90	284.17	286.34	3.01
total	480	461	104 (1/4.6x)(1/4.4x)	—	—	—	—	—	—
average	—	—	—	8.45	8.46	8.51	46.05	47.04	10.94 (1/4.2x)(1/4.3x)

fact, D-Router reduced wire length slightly after manually routing all nets due to its higher routability.

The runtime of an initial routing of D-Router is also added into the runtime as a whole in this way, D-Router can be viewed as an identical functional module for comparison with the two alternative algorithms. The initial routing actually takes only one iteration of algorithm [7] without any congestion constraint. [7] and negotiation-based routing have similar runtime, and D-Router reduces runtime by on an average 4.3x but up to 94x compared to the two alternative algorithms for the large-scale example S10, since test case S10 is a SIP (System-in-Package) package with multiple dies in one package. As for the two alternative algorithms, such a problem requires more dynamic searching steps [7], and even more steps when a net fails to route which searches all the possible triangles. Furthermore, more iterations are also required for convergence. However, the initial routing of D-Router doesn't consider congestion constraint, thus a routing solution is easily found. The congestion diffusion process has the same complexity with other cases. Therefore, both alternative algorithms take much longer runtime than the D-router, especially for test case S10. This demonstrates the quality of D-Router most effectively.

5. CONCLUSIONS

We have developed a diffusion-based topological router (D-Router) for congestion reduction in substrate routing. Experiments using industrial design examples show that a very recent substrate routing method [7] leaves 480 nets unrouted for ten industrial designs with a total of 6415 nets, while D-Router reduces the number of unrouted nets to 104, a 4.6x net number reduction, which translates to substantial design time reduction. Our algorithm also reduces runtime by an average 4.3x but up to 94x.

While the negotiation based routing algorithm obtains good on-chip routing results for both FPGA and ASIC [3, 8, 9], results in this paper show that D-Router significantly outperforms negotiation-based algorithm in terms of both routability and runtime in a package substrate routing problem. Although D-Router is used for substrate routing in this paper, its concept can be applied to on-chip routing as well. Extending D-Router to on-chip routing will be our future work.

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] M. Cho, K. Lu, K. Yuan, and D. Pan. BoxRouter 2.0:architecture and implementation of a hybrid and robust global router. In *Proc. Intl. Conf. Computer-Aided Design*, pages 503–508, 2007.

[4] M. Cho and D. Pan. BoxRouter: A new global router based on box expansion and progressive ILP. In *Proc. Design Automation Conf.*, pages 373–378, 2006.

[5] W. W. Dai, T. Dayan, and D. Staepelaere. Topological routing in SURF: generating a rubber-band sketch. In *Proc. Design Automation Conf.*, pages 39–44, 1991.

[6] 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.

[7] S. H. Liu, G. Q. Chen, T. T. Jing, L. He, T. P. Zhang, R. Dutta, and X. L. Hong. Topological routing to maximize routability for package substrate. In *Proc. Design Automation Conf.*, pages 566–569, 2008.

[8] L. McMurchie and C. Ebeling. PathFinder: a negotiation-based performance-driven router for FPGAs. In *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, pages 111–117, 1995.

[9] J. Roy and I. Markov. High-performance routing at the nanometer scale. In *Proc. Intl. Conf. Computer-Aided Design*, pages 496–502, 2007.

[10] R. Shi and C. K. Cheng. Efficient escape routing for hexagonal array of high density I/Os. In *Proc. Design Automation Conf.*, pages 1003–1008, 2006.

[11] 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.

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

[13] J. J. Xiong, Y. C. Wong, E. Sarto, and L. He. Constraint driven I/O planning and placement for chip-package co-design. In *Proc. Asia South Pacific Design Automation Conf.*, pages 207–212, 2006.

[14] 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.

[15] 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.