BoxRouter 2.0: A Hybrid and Robust Global Router with Layer Assignment for Routability

MINSIK CHO, KATRINA LU, KUN YUAN, and DAVID Z. PAN
The University of Texas at Austin

In this article, we present BoxRouter 2.0, and discuss its architecture and implementation. As high-performance VLSI design becomes more interconnect-dominant, efficient congestion elimination in global routing is in greater demand. Hence, we propose a global router which has a strong ability to improve routability and minimize the number of vias with blockages, while minimizing wirelength. BoxRouter 2.0 is extended from BoxRouter 1.0, but can perform multi-layer routing with 2D global routing and layer assignment. Our 2D global routing is equipped with two ideas: node shifting for congestion-aware Steiner tree and robust negotiation-based A* search for routing stability. After 2D global routing, 2D-to-3D mapping is done by the layer assignment which is powered by progressive via/blockage-aware integer linear programming. Experimental results show that BoxRouter 2.0 has better routability with comparable wirelength than other routers on ISPD07 benchmark, and it can complete (no overflow) the widely used ISPD98 benchmark for the first time in the literature with the shortest wirelength. We further generate a set of harder ISPD98 benchmarks to push the limit of BoxRouter 2.0, and propose the hardened ISPD98 benchmarks to map state-of-the-art solutions for future routing research.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuit**]: Design Aids

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: VLSI, physical design, global routing, congestion, routability, layer assignment, integer linear programming

ACM Reference Format:

Cho, M., Lu, K., Yuan, K., and Pan, D. Z. 2009. BoxRouter 2.0: A hybrid and robust global router with layer assignment for routability. ACM Trans. Des. Autom. Elect. Syst., 14, 2, Article 32 (March 2009), 21 pages, DOI = 10.1145/1497561.1497575 http://doi.acm.org/10.1145/1497561.1497575

1. INTRODUCTION

While ever-decreasing feature size enables the integration of millions of gates on a chip, interconnect delay becomes the dominant factor in VLSI performance

This work is supported in part by NSF, SRC, IBM Faculty Award, Fujitsu, Sun, and equipment donations from Intel.

Authors' addresses: M. Cho, K. Lu, K. Yuan, and D. Z. Pan; Electrical and Computer Engineering Department, University of Texas at Austin, Austin, TX 78712; email: {thyeros,yiotse,kyuan}@cerc.utexas.edu; dpan@ece.utexas.edu.

DOI 10.1145/1497561.1497575 http://doi.acm.org/10.1145/1497561.1497575

[Semiconductor Industry Association 2007; Cong 1997; Kastner et al. 2001; Wu et al. 2005]. Thus, every stage of design process targets for minimal wirelength to enhance circuit delay. Especially placement, a major step in physical design, generally minimizes wirelength by placing gates more compactly. In addition, more functionalities in complex design (i.e., system-on-chip) also demand more gates in a limited die, consequently increasing design density. Such design trends degrade routability by leaving the design with limited routing area and thus make wiring gates more challenging. Therefore, routability should be one of the most critical design objectives in VLSI physical design [Hu and Sapatnekar 2000; Hu and Sapatnekar 2002; Westra et al. 2005].

Routability can be enhanced in multiple stages in physical design [Kutzschebauch and Stok 2001; Wenting et al. 2001; Brenner and Rohe 2003], but routing is the most effective stage, as it plans wire distribution and embeds wires under design rules with the accurate pin and blockage information in hand. Routing consists of two steps, global routing and detailed routing. Global routing plans an approximate path for each net, while detailed routing finalizes the exact DRC-compatible pin-to-pin connections. As detailed routing cannot capture overall congestion due to fine routing grid size and numerous design rules, global routing should eliminate congestion by migrating wires from more to less congested regions with the minimized overhead in wirelength and via. If global routing fails to satisfy congestion constraints, it will incur significant design cost, as a chip should be resynthesized not to have any congestion before tape out. Therefore, routability should be the primary goal of global routing.

The significance of routability in VLSI global routing has led to many global routing algorithms. Burstein and Pelavin [1983] proposed a hierarchical approach to speed up an integer programming formulation for global routing, and Kastner et al. [2002] proposed pattern-based global routing. Hadsell and Madden [2003] presented *Chi* dispersion router based on a linear cost function as well as a predicted congestion map, and showed better results than Kastner et al. [2002]. The multicommodity flow-based global router by Albrecht [2001] showed good results and was used in industry, but at the expense of computational effort. After BoxRouter [Cho and Pan 2006] sparked the renewed interest in routing research with significantly improved performance, FastRoute [Pan and Chu 2006, 2007] and DpRouter [Cao et al. 2007] achieved high quality solutions in small runtime. However, most of the academic global routers work in 2D (with two layers) to handle a larger circuit with less computing power and smaller memory, and lack the important layer assignment.

Layer assignment plays critical roles for routability, timing, crosstalk, and manufacturability/yield. If an excessive number of wires are assigned to a particular layer, it will aggravate congestion and crosstalk [Kay and Rutenbar 2000; Wu et al. 2004]. If global timing critical nets are assigned to lower layers, it will make timing worse due to narrower wire width/spacing. Biased wire density distribution between layers can cause large topography variation as well as pooling effect after CMP [Cho et al. 2006]. Length of antenna can be also reduced by layer assignment [Wu et al. 2006]. A large number of vias due to poor layer assignment can cause routability/pin access problems, as via (even extended via) needs larger area as well as wider spacing than wire. Especially,

via minimization becomes more important for nanometer design due to manufacturability. With a smaller number of vias, we can decrease the chance of via failure by increasing the percentage of redundant via insertion [Xu et al. 2005; Lee and Wang 2006].

Recent global routing contest in ISPD-2007[1] attracted 17 teams from both academia and industry, reflecting the renaissance of routing. It provided 16 industrial benchmarks (8 for 2D and another 8 for 3D) to emphasize the importance of routability in global routing and the necessity of via minimization in layer assignment. The contest results in several highly advanced academic global routers based on the history idea originally proposed in McMurchie and Ebeling [1995]. FGR [Roy and Markov 2007] is based on Lagrangian relaxation and Steiner tree reconstruction to achieve high quality solutions. ARCHER [Ozdal and Wong 2007] adopts also a Lagrangian relaxation technique with congestion history learning, achieving high quality and fast runtime. NTHU-Route [Gao et al. 2008] uses a similar history-based routing technique. Differently from these history-based (broader context of Lagrangian relaxation) global routers, MaizeRouter relies on efficient edge shifting to improve routability.

In this work, we propose another global router, BoxRouter 2.0 which consists two steps, 2D global routing and layer assignment. 2D global routing boasts strong routability based on two techniques, namely node shifting and robust negotiation-based A* search. Meanwhile, layer assignment is enabled by novel and efficient progressive via/blockage-aware integer linear programming (ILP). The major contributions of this article include the following:

—We propose a node shifting technique to refine congestion-aware Steiner tree, and show that node shifting combined with edge shifting [Pan and Chu 2006] is more effective than edge shifting only for hard cases.

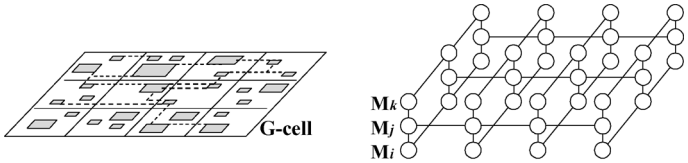
—We propose simple, yet essential dynamic scaling for robust negotiation-based A* search. This prevents a router from spinning out of control by balancing historic cost and present congestion cost, and ensures consistent routability improvement over iterations.

—We propose integer linear programming (ILP) for via/blockage-aware layer assignment to handle blockages and guarantee the feasibility. Also, we apply a progressive ILP technique to via/blockage-aware layer assignment in order to enhance runtime.

—We complete ISPD98 benchmarks without any overflow in the shortest wirelength for the first time, compared with all published academic global routers. Also, BoxRouter 2.0 finishes the most of number of circuits with comparable wirelength on ISPD07 global routing benchmarks, compared with all winning global routers.

—We propose two modified ISPD98 benchmarks with significantly reduced routing capacities to push the limit of global routers, and report the results of BoxRouter 2.0 on these as a reference for future research.

[1]http://www.ispd.cc/ispd07_contest.html



(a) real circuit with G-cells (b) grid graph for routing

Fig. 1. A circuit with netlists can be dissected into multiple grids which can be mapped into graph for global routing.

The rest of the article is organized as follows. Section 2 presents preliminaries. Section 3 provides an overview of BoxRouter 2.0. Details on our 2D global routing is described in Section 4, then layer assignment is proposed in Section 5. Experimental results are discussed in Section 6, followed by conclusion in Section 7.

2. PRELIMINARIES

2.1 Global Routing Background

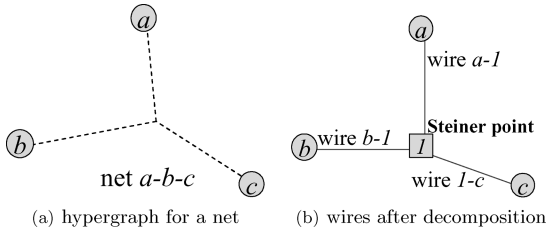
The global routing problem can be modeled as a grid graph search problem. Once a circuit is partitioned into a set of rectangular regions as shown in Figure 1(a), we represent each region of the circuit by the same number of vertices as the number of metal layers in the given manufacturing process as in Figure 1(b). Each metal layer is dedicated to either horizontal or vertical wires. A vertex is called a global routing cell (G-cell), and each edge represents the boundary between G-cells. Each edge has maximum routing capacity, and each wire passing the edge takes some routing capacity based on its width/spacing. When the demand from wires exceeds the maximum routing capacity of the edge, overflow occurs. The number of overflow can be computed as the excessive demand [Westra et al. 2005; Kastner et al. 2002], which is a commonly agreed metric for routability. Thus, a global routing is to find paths that connect the pins inside the G-cells through the graph for every net with fewer overflows [Westra et al. 2005]. Since a net may have a complex topology, it can be decomposed into two pin *wires* with Rectilinear Minimum Steiner Tree [Cho and Pan 2006; Pan and Chu 2006; Hentschke et al. 2007].

2.2 Net Decomposition

A net can be decomposed into two pin *wires* with Rectilinear Minimum Steiner Tree as shown in Figure 2, where a net *a*-*b*-*c* is decomposed into three wires by the Steiner point 1. After net decomposition, there are two kinds of wires, one is flat wire like wire *b*-1, and the other is one-bend wire like wire 1-*c*. Each wire will be considered as one routing objects; routing is done in a wire-by-wire manner.

2.3 BoxRouter 1.0

BoxRouter 1.0 [Cho and Pan 2006, 2007] is based on congestion-initiated box expansion; it progressively expands a box which initially covers the most congested region only, but finally covers the whole circuit. Within each box,



(a) hypergraph for a net (b) wires after decomposition

Fig. 2. A net can be decomposed into two pin wires with Rectilinear Minimum Steiner Tree Construction.

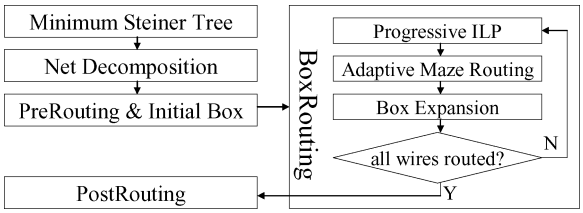


Fig. 3. BoxRouter 1.0 overall flow.

BoxRouter 1.0 performs progressive integer linear programming (ILP) and adaptive maze routing to effectively diffuse the congestion as in Figure 3. To decide the first box based on the global congestion view, BoxRouter 1.0 performs PreRouting. After all nets are routed, PostRouting further improves the solution by rerouting detoured nets. BoxRouter 1.0 [Cho and Pan 2006] shows significantly superior results on ISPD98 benchmark, compared with Hadsell and Madden [2003], Kastner et al. [2002], and Albrecht [2001].

However, BoxRouter 1.0 has one limitation for highly congested designs where one general assumption of global routing (i.e., 70%–80% of nets are destined to be routed in a simple L-shape pattern [Westra et al. 2004, 2005]) does not hold. In detail, its progressive ILP formulation for routing only considers the L-shape pattern based on such assumption, but it does not work well for hard cases where most nets need to be detoured in complicated patterns. However, considering various routing patterns in ILP is prohibitively expensive due to the increase in the number of variables in ILP.

2.4 Negotiation-Based Routing

It is shown that negotiation-based routing is effective in congestion elimination for FPGA [McMurchie and Ebeling 1995] as well as ASIC [Cong et al. 2005]. The key idea of the negotiation-based approach is that the congestion history of every edge in the routing graph will be considered for the future routing. In detail, for each edge e, there are two cost factors: $h^i(e)$ for historic cost at the i-th iteration and $p(e)$ for the present congestion cost. The combination of these two factors will provide the final cost for a wire to pass through e. As $h^i(e)$ is increased for any congested edge e right after each iteration, an edge which has been congested previously tends to have high $h^i(e)$. Meanwhile, $p(e)$ is solely related to the present congestion of e. Thus, considering both $h^i(e)$ and $p(e)$ as

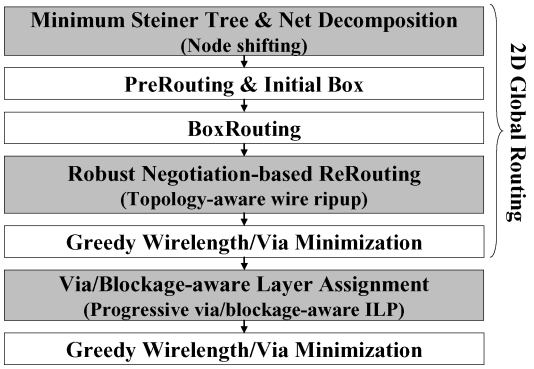


Fig. 4. The overview of BoxRouter 2.0.

routing cost will guide a router to avoid the presently congested edges as well as previously congested edges. This is a very efficient technique to spread out wires to less congested regions.

3. OVERVIEW OF BOXROUTER 2.0

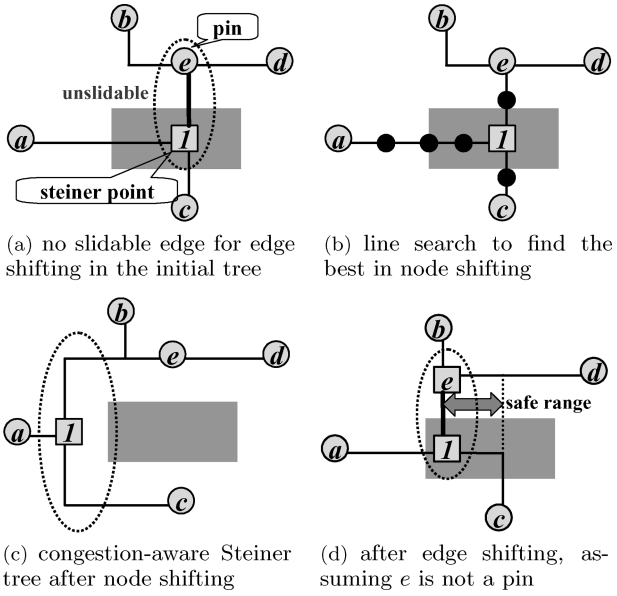
In this section, we give the overview of BoxRouter 2.0 shown in Figure 4. The early steps of BoxRouter 2.0 are inspired by BoxRouter [Cho and Pan 2006], but ours is radically different in a sense that we have a more powerful and systematic way of removing congestion and assigning layers to wires. BoxRouter 2.0 has two major steps, 2D global routing (Section 4) and layer assignment (Section 5). When a circuit to route is given, we superpose all the layers into two layers, the horizontal and vertical, then perform 2D global routing to maximize routability. Layer assignment follows 2D global routing to distribute wires across multiple layers, while minimizing the number of vias.

In fact, our 2D global routing can be applied for multiple layers (3D) directly, but the advantage of 2D global routing over 3D global routing is that it needs less computing power and memory, as the global routing graph shrinks significantly. Also, the mapping from 2D solution to 3D solution can be done without making congestion worse, as long as a wire can be splitted to avoid blockages at a cost of via and wire width/spacing are regarded as constant.

4. 2D GLOBAL ROUTING

In this section, we present our 2D global routing algorithm. As BoxRouter 2.0 is inspired by PreRouting and BoxRouting of BoxRouter [Cho and Pan 2006], we take them to generate an initial routing solution as in Figure 3. However, we apply our node shifting technique to make Steiner tree more congestion-aware for hard cases, and improve routability considerably by our negotiation-based A* search. Our technical contributions in 2D global routing can be summarized as follows:

(1) *Node shifting.* In congestion-aware Steiner tree construction, a Steiner point can be shifted to a less congested region to reduce the overall congestion. This is in Section 4.1.



(a) no slidable edge for edge shifting in the initial tree

(b) line search to find the best in node shifting

(c) congestion-aware Steiner tree after node shifting

(d) after edge shifting, assuming e is not a pin

Fig. 5. Node shifting can complement edge shifting, if a high degree Steiner point is inside congestion.

(2) *Robust negotiation-based A* search.* This is an important idea to enable continuous and *stable* routability improvement during whole rerouting procedure as discussed in Section 4.2.

4.1 Node Shifting for Steiner Tree

When we generate Rectilinear Minimum Steiner Tree as our starting point of the routing problem, the solution may not be good with respective to congestion, although we can achieve near-optimal wirelength. This motivates the congestion-aware Steiner tree. Edge shifting is proposed in Pan and Chu [2006] as a pioneering technique to make the Steiner tree congestion-aware by sliding wires to less congested regions. However, edge shifting is not effective for some cases, where a Steiner point is trapped in the congested region. Thus, we propose a node shifting technique to complement edge shifting and refine congestion-aware Steiner tree further.

To compare node shifting with edge shifting, consider the example in Figure 5, where pins are in circle (a,b,c,d,e) and a Steiner point is in square (1). The initial Steiner tree in Figure 5(a) has the Steiner point 1 inside congestion in the dark area. As edge shifting slides an edge to a less congested region only if both of its end points are Steiner points, it cannot do any optimization for Figure 5(a) (there is no *slidable* edge for edge shifting). Whereas node shifting performs line search as shown in Figure 5(b), then shifts the Steiner point 1 to the best candidate, resulting in Figure 5(c). Even if we consider the case in Figure 5(d) where e is a Steiner point (e is now in square) to have one slidable edge, the net will still be in congestion due to narrow safe range [Pan and Chu 2006]. The key advantage of node shifting over edge shifting is that it provides

larger flexibility by allowing bigger radical changes in the topology. However, as this involves overhead in wirelength and via, when searching for the best candidate, cost and benefit should be considered together.

A similar idea on node shifting is proposed by Alpert et al. [2004] for buffer insertion, but different from our node shifting in terms of search space. The approaches in these works relocate a Steiner point to a new location within a two-dimensional *plate* to modify the tree topology according to the congestion. While the plate needs two-dimensional search for the new location, our proposed approach need one dimensional line search along the current tree edges, which can be much faster for most cases. Searching two-dimensional space can provide larger flexibility, but it can be computationally expensive for congestion-aware Steiner tree construction, a preliminary step of global routing. Also, plate-based search requires a well-tuned cost function to avoid highly suboptimal new locations.

We observe that node shifting would not be useful, if a design is quite low congested considering the strength of a router. All the expected advantages from node shifting for sparse designs can be washed out during routing, but the overhead due to congestion-awareness can negatively affect the routing quality in terms of wirelength and via. Hence, node shifting is applied only for the highly congested design.

4.2 Robust Negotiation-Based A* Search

Instead of maze routing/shortest path algorithm, we adopt A^* search algorithm and use the following cost function in BoxRouter 2.0.

$$cost^i(e) = h^i(e) + \alpha p(e) + \beta d(e), \tag{1}$$

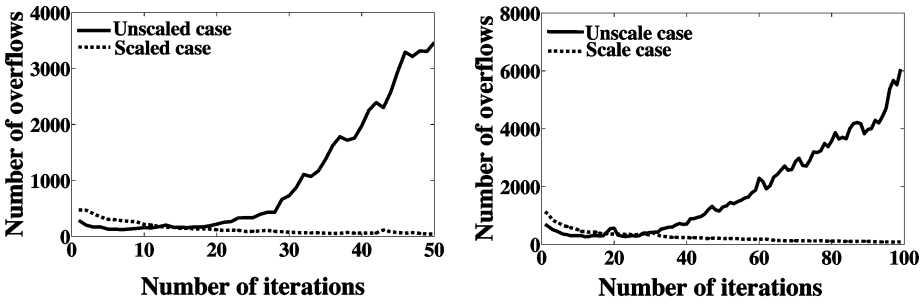
where regarding an edge e, $h^i(e)$ is a historic cost at i-th iteration, $p(e)$ is a present congestion cost, and $d(e)$ is the distance from e to the target. In detail, $p(e)$ can be computed as follows where $C(e)$ and $U(e)$ denote the total capacity and used capacity of e respectively.

$$p(e) = P\frac{U(e)}{C(e)}, \tag{2}$$

where P indicates the congestion cost when there is no available routing capacity in the edge e. Also, $h^i(e)$ can be computed as follows.

$$h^i(e) = \begin{cases} h^{i-1}(e) & p(e) \le P \\ h^{i-1}(e) + i & p(e) > P. \end{cases} \tag{3}$$

We find that there can be a potential stability problem with negotiation-based A* search for highly congested designs which need a large number of iterations. For every iteration, $h^i(e)$ is increased, if e is congested. Thus, after many iterations which frequently happens for highly congested designs, $h^i(e)$ starts to dominate $p(e)$. This implies that a presently congested edge becomes cheaper to pass through than a previously congested edge. This may lead to routing instability in a sense that the solution quality may get worse with more iterations due to the unbalance between $h^i(e)$ and $p(e)$. Thus, to ensure



(a) ibm01 (b) ibm04

Fig. 6. Dynamically scaled A* search reduces congestions robustly and stably over iterations.

continuous improvement in routability, the balance between two costs has to be kept.

To address this instability problem and make a router robust, we scale $p(e)$ by picking the following α for Equation (1).

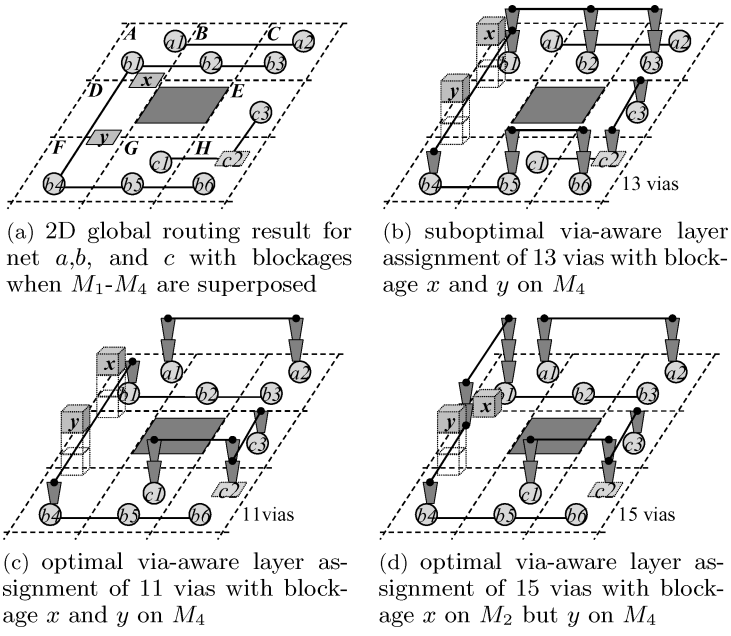
$$\alpha = \frac{max_e[h^i(e)]}{P}. \tag{4}$$

Insight behind such α is to make a presently congested edge (no more routing capacity available) passing as expensive as a previously congested edge passing. Therefore, our scaling factor is not static but changes dynamically each iteration. This will discourage creating new overflows, while avoiding previously congested edges.

Figure 6 shows the effect of robust negotiation-based A* search by comparing the scaled case (Equation (4)) and unscaled case ($\alpha = 1$) on two benchmark circuits. For the unscaled case, it reduces the overflows faster than the scaled case for a while, but after a certain point, it spins a router out of control and increases the number of overflows. This implies that if circuit is too hard to be routed in a few iterations, a router becomes so unstable that it cannot improve the routing quality. Meanwhile, the scaled case stably reduces the number of overflows even after a large number of iterations. With larger fixed/unscaled α, we may delay spinning out of control, but it will eventually occur after a larger number of iterations.

5. LAYER ASSIGNMENT

In this section, we propose a layer assignment for via-minimization based on progressive integer linear programming (ILP). When 2D global routing is finished, layer assignment follows to distribute the wires across the layers. Layer assignment impacts several design objectives, such as timing, noise, and manufacturability, but our layer assignment mainly focuses on via minimization without altering any routing topology. This problem is known as constrained via minimization (CVM) [Chang and Du 1988; Chang and Cong 1999; Ahn and Sahni 1993] which is shown to be NP-complete [Naclerio et al. 1989]. However, our layer assignment for via minimization inherently differs from previous works on CVM in two aspects.



(a) 2D global routing result for net a,b, and c with blockages when M_1-M_4 are superposed

(b) suboptimal via-aware layer assignment of 13 vias with blockage x and y on M_4

(c) optimal via-aware layer assignment of 11 vias with blockage x and y on M_4

(d) optimal via-aware layer assignment of 15 vias with blockage x on M_2 but y on M_4

Fig. 7. Layer assignment can determine the number of vias as shown in (b) and (c). Also, the location of blockages in 3D can affect routability in (d).

—multiple wires can be overlapped, if there are enough routing capacities between G-cells, while CVM does not allow due to DRC.

—wires can be splitted into multiple pieces to avoid blockages, while CVM cannot.

Integer linear programming also has been used for layer assignment in several previous works [Ciesielski and Kinnen 1981; Shi et al. 1997; Jhang 2000], but ours is more applicable to large scale VLSI design, because we assume neither a fixed number of layers nor the feasibility of a layer assignment problem.

5.1 Via-Aware Layer Assignment

Depending on layer assignment, the number of vias can be significantly different. Figure 7 shows an example of layer assignment for via minimization, where net a, b, and c are routed through 2D global routing cells, and pins are shown in circle, while a bend ($c2$) in square. The example assumes four metal layers (M_1-M_4), where M_1 and M_3 are for horizontal wires, M_2 and M_4 are for vertical wires, and all the pins on M_1. Further, a single routing capacity is assumed for each edge. If a greedy approach (a shorter net is assigned to a lower layer) is adopted, it will result in Figure 7(b) with 13 vias. However, Figure 7(b) has 2 more vias (18%) than the optimal assignment in Figure 7(c). This is simply because the greedy approach cannot capture the global view. Hence, we propose an integer linear programming (ILP) formulation for via-aware layer assignment shown in Figure 8 where notations are in Table I.

$$
\begin{array}{lll}
\textbf{min :} & & \sum_i \sum_{s \in P(i)} (T_{is} - B_{is}) \\
\textbf{s.t :} & (a) & k \in \{M_i | 1 \le i \le n\} \\
& (b) & z_{ijk} \in \{0,1\} \qquad \forall i,j,k \\
& (c) & \sum_k z_{ijk} = 1 \qquad \forall i,j,k \\
& (d) & \sum_k k \cdot z_{ijk} = l_{ij} \qquad \forall i,j,k \\
& (e) & B_{is} \le l_{ij} \le T_{is} \qquad \forall (i,j) \in W(i,s) \\
& (f) & B_{is} = M_1 \qquad \forall s \in N(i) \\
& (g) & \sum_{(i,j,k) \in C(e)} z_{ijk} \le r_e \qquad \forall e \\
& (h) & M_i < M_{i+1} \qquad 1 \le i \le n-1
\end{array}
$$

Fig. 8. ILP formulation for via-aware layer assignment.

Table I. The Notations in Figures 8–10

$W(i,s)$	a set of wires of a net i passing a point s (including pins)
$P(i)$	a set of points in a net i
$N(i)$	a set of pins in a net i ($N(i) \subseteq P(i)$)
$C(e)$	a set of wires crossing an edge e
r_e	the available routing capacity of an edge e
z_{ijk}	a binary variable set to 1 if a wire j of a net i is assigned k layer
l_{ij}	the layer assigned to a wire j of a net i
T_{is}	the top layer assigned to any wire on a point $s \in P(i)$
B_{is}	the bottom layer assigned to any wire on a point $s \in P(i)$

The objective is to minimize the difference between the top layer and bottom layer used by wires of each net for each point. First, the constraint (a) defines a set of layers eligible for assignment where n is the maximum layer number for routing. The constraint (c) is to assign a wire j of a net i to one of the layers. l_{ij} of the constraint (d) is computed by the combination of z_{ijk}. Then, T_{is} and B_{is} are captured by the constraint (e). If there is a pin on s, the B_{is} is set as M_1 in the constraint (f). Finally, all the layer assignment cannot exceed the capacities of all the edges by the constraint (g). Figure 9 shows the ILP formulation for Figure 7(a), mainly focusing on the net c. Although the proposed ILP formulation can optimally minimize the number of vias during layer assignment, it has two drawbacks:

—Depending on blockage locations, the formulation can be infeasible, which will be addressed in Section 5.2.

—ILP inherently cannot be applied for large designs. Thus, it needs a technique to improve the speed, which will be discussed in Section 5.4.

5.2 Via/Blockage-Aware Layer Assignment

Since the exact layer information on blockages is diluted in 2D global routing, the layer assignment based on the 2D routing result may not be feasible. Compare Figure 7(c) and Figure 7(d), where the blockage x is located in different layers. In Figure 7(c), both x and y are on M_4, enabling to route wire $b1 - b4$ on M_2. However, in Figure 7(d), wire $b1 - b4$ cannot be routed *as it is*, as x is

$$
\begin{array}{rll}
\textbf{min :} & \sum_{s\in\{c1,c2,c3\}}(T_{cs}-B_{cs})+\sum_{i\in\{a,b\}}\sum_{s\in P(i)}(T_{is}-B_{is}) & \\
\textbf{s.t :} & z_{c1-c2,M_1}, z_{c1-c2,M_3} & \in\{0,1\}\\
& z_{c2-c3,M_2}, z_{c2-c3,M_4} & \in\{0,1\}\\
& z_{c1-c2,M_1}+z_{c1-c2,M_3} & =1\\
& z_{c2-c3,M_2}+z_{c2-c3,M_4} & =1\\
& M_1\cdot z_{c1-c2,M_1}+M_3\cdot z_{c1-c2,M_3} & =l_{c1-c2}\\
& M_2\cdot z_{c2-c3,M_2}+M_4\cdot z_{c2-c3,M_4} & =l_{c2-c3}\\
& B_{c1}\le \quad l_{c1-c2} \quad \le T_{c1} & \\
& B_{c2}\le l_{c1-c2}, l_{c2-c3}\le T_{c2} & \\
& B_{c3}\le \quad l_{c2-c3} \quad \le T_{c3} & \\
& B_{c1}, B_{c3} & =M_1\\
& z_{c1-c2,M_1}+z_{b5-b6,M_1} & \le 1\\
& z_{c1-c2,M_3}+z_{b5-b6,M_3} & \le 1\\
& z_{c2-c3,M_2} & \le 1\\
& z_{c2-c3,M_4} & \le 1\\
& constraints \quad for \quad a \quad and \quad b \quad ... &
\end{array}
$$

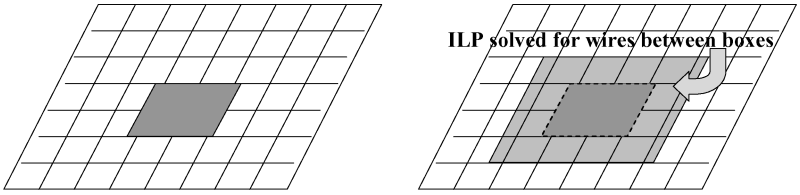
Fig. 9. Example of ILP formulation for via-aware layer assignment for Figure 7(a).

$$
\begin{array}{rll}
\textbf{min :} & \sum_i\sum_{s\in P(i)}(T_{is}-B_{is})-\alpha\sum_{i,j,k}z_{ijk} \quad (\alpha\gg 1) & \\
\textbf{s.t :} & k\in\{M_i|1\le i\le n\} & \\
& z_{ijk}\in\{0,1\} & \forall i,j,k\\
& \sum_k z_{ijk}\le 1 & \forall i,j,k\\
& \sum_k k\cdot z_{ijk}=l_{ij} & \forall i,j,k\\
& B_{is}\le l_{ij}\le T_{is} & \forall (i,j)\in W(i,s)\\
& B_{is}\le M_1 & \forall s\in N(i)\\
& \sum_{(i,j,k)\in C(e)} z_{ijk}\le r_e & \forall e\\
& M_i<M_{i+1} & 1\le i\le n-1
\end{array}
$$

Fig. 10. ILP formulation for via/blockage-aware layer assignment.

on M_2 while y is on M_4. Wire $b1-b4$ should be chopped into two pieces such that it can shuttle from M_2 to M_4 as in Figure 7(d). Thus, unless wire $b1-b4$ is splitted, the formulation in Figure 8 becomes infeasible. This issue can be easily addressed by chopping wires, wherever a blockage exits, but this may result in not only unnecessary vias but also too many variables in ILP. Therefore, it is better to break a wire only if needed.

Motivated by the idea in Cho and Pan [2006], we propose a new ILP formulation for via/blockage-aware layer assignment as shown in Figure 10, where the constraint (b) in Figure 8 is relaxed, and the objective is modified. This formulation is guaranteed to be feasible for any blockage distribution. In fact, the new formulation does not require layer assignment for all wires, but the objective is to complete as many wires as possible, while minimizing the number of vias. The unassigned wires after solving ILP will be picked up by a maze routing like Cho and Pan [2006]. But, differently from Cho and Pan [2006], our maze routing is much simpler and faster, because it only needs to shuttle



(a) after i-th iteration, all the wires are layer-assigned by ILP of Fig. 10 followed by simple maze routing.

(b) in i+1-th iteration, the wires between two boxes are to be layer-assigned in the same manner.

Fig. 11. Progressive ILP based on box expansion is efficient in managing problem size tractable, while honoring the solutions from previous iterations.

between layers to finish assignment. Also, we assign high penalty on each layer change to minimize the number of vias during maze routing. Therefore, fewer wires will be chopped than the approach of chopping wires before solving ILP, resulting in fewer vias in shorter runtime.

5.3 Implementation Issues

While implementing the formulation in Figure 10, the layer number k (e.g., $M_1, M_2, \dots$) should be defined as a positive number. The reason is because when $z_{ijk} = 0$ for some (i, j), $l_{ij} = 0$ (implying no wire is assigned), which should be distinguished from valid layer numbers like M_1, M_2, and so on. Finally, $M_{i+1} - M_i, (1 \leq i \leq n-1)$ needs to be constant to minimize the number of vias. Otherwise, each via at different layers will have different weights in the cost function, resulting in a suboptimal solution.

5.4 Progressive ILP for Via/Blockage-Aware Layer Assignment

ILP is computationally expensive, as most solvers use branch-and-bound algorithms. Thus, in order to apply ILP to industrial designs, the problem size should be tractable, while maintaining the global view. We adopt the idea of box expansion and progressive ILP formulation [Cho and Pan 2006] for our layer assignment. Figure 11 illustrates the core idea of progressive ILP. It starts with a minimal box covering the most congested region. Then, we solve the problem (in our case, layer assignment) inside the box by ILP as in Figure 11(a). After the box expands to cover the larger area, the problem inside the expanded box is solved in the way as shown in Figure 11(b). However, since the current problem encloses the previous problem (which has been solved), the actual problem is limited down to layer assignment of only the wires between two consecutive boxes, which in turn makes the problem size tractable. Additionally, the previous solution becomes a part of the current problem, thus all the decisions made previously are honored by the current optimization.

6. EXPERIMENTAL RESULTS

We implement BoxRouter 2.0 in C++, and perform all the experiments on 2.8 GHz Pentium 32bit Linux machine with 2GB RAM. Congestion-aware Steiner tree construction [Pan and Chu 2006] based on Flute [Chu 2004] is

Table II. ISPD07 IBM Benchmarks

name[a]	nets	grids	v.cap[b]	h.cap[b]	placer
adaptec1	219794	324 × 324	70	70	Capo
adaptec2	260159	424 × 424	80	80	mPL6
adaptec3	466295	774 × 779	62	62	Dragon
adaptec4	515304	774 × 779	62	62	APlace3
adaptec5	867441	465 × 468	110	110	mFAR
newblue1	331663	399 × 399	62	62	NTUplace 3.0
newblue2	463213	557 × 463	110	110	FastPlace 3.0
newblue3	551667	973 × 1256	80	80	Kraftwerk

[a]2D cases have 2 layers, but 3D cases have 6 layers.
[b]vertical/horizontal capacity

adopted. We use ISPD07 benchmarks to demonstrate BoxRouter 2.0. Also, we apply BoxRouter 2.0 to ISPD98 benchmarks as well, and further modify ISPD98 benchmarks to see the limit of BoxRouter 2.0. Details on ISPD07 and ISPD98 benchmarks are presented in Table II and V respectively.

6.1 ISPD07 Benchmarks

We report the results of all the winners (FGR 1.0, MaizeRouter, and BoxRouter 1.9) of ISPD-2007 routing contest,[2] FGR 1.1 [Roy and Markov 2007] and ARCHER [Ozdal and Wong 2007], and BoxRouter 2.0 on ISPD07 benchmarks in Table III. Regarding wirelength, ours is similar to or better than BoxRouter 1.9, MaizeRouter, ARCHER (especially for 3D benchmark), and comparable with FGRs. However, BoxRouter 2.0 completes 12 designs, which ties with BoxRouter 1.9, ARCHER, and FGR 1.1. Additionally, BoxRouter 2.0 overall achieves fewer total and maximum overflows, which may be easily fixed during detailed routing. All the results prove that BoxRouter 2.0 has the state-of-the-art routability, which is the utmost goal of global routing, and provides high quality solutions in terms of wirelength/via. For ISPD-2007 contest, the runtime was not a metric, but our runtime for the biggest circuit is about 2 days.

6.2 ISPD98 Benchmarks

We use ISPD98 benchmarks to compare BoxRouter 2.0 with recently published global routers, *Chi* Dispersion, BoxRouter 1.0, ARCHER, FGR 1.0, and FastRoute 2.0. Table IV shows the performance of each router on ISPD98 benchmarks. We normalize the numbers by those from FastRoute 2.0, as it has been the best in the literature. First, it shows that BoxRouter 2.0 completes ISPD98 benchmarks without any overflow. We tune BoxRouter 2.0 for runtime and quality (which is the default setting) respectively, and compare both results with other global routers as shown in Table IV. When tuned for runtime, although slower than FastRoute 2.0, ours is 2-12x faster than the others, and better congestion distribution (no overflow) will be significantly rewarded in detailed routing by huge speed-up. Therefore, a higher quality solution should

[2]http://www.ispd.cc/ispd07_contest.html

Table III. Comparison Between ISPD07 Contest Awardees/ICCAD07 Publications and Ours on ISPD07 Benchmarks

2D

name	BoxRouter 1.9			FGR 1.0			MaizeRouter			ARCHER[d]			FGR 1.1			BoxRouter 2.0		
	wlen[a]	max.o[b]	ovfl[c]	wlen	max.o	ovfl	wlen	max.o	ovfl	wlen	ovfl	wlen	max.o	ovfl	wlen	max.o	ovfl
adaptec1	58.84	0	0	55.8	0	0	62.26	0	0	57.66	0	53.81	0	0	58.37	0	0
adaptec2	55.69	0	0	53.69	0	0	57.23	0	0	54.62	0	51.86	0	0	55.69	0	0
adaptec3	140.87	0	0	133.34	0	0	137.75	0	0	135.18	0	129.58	0	0	137.96	0	0
adaptec4	128.75	0	0	126.05	0	0	128.45	0	0	126.32	0	124.12	0	0	127.79	0	0
adaptec5	164.32	0	0	155.82	0	0	176.69	2	2	162.71	0	150.64	0	0	162.11	0	0
newblue1	51.13	2	400	47.51	10	1.2K	50.93	16	1.3K	48.74	494	47.43	4	452	51.13	2	400
newblue2	79.78	0	0	77.67	0	0	79.64	0	0	77.94	0	75.87	0	0	78.68	0	0
newblue3[d]	111.64	1.1K	39K	108.18	1.1K	37K	114.63	1.2K	33K	10959	32K	109.34	1120	39K	111.61	1.1K	39K

3D

name	BoxRouter 1.9			FGR 1.0			MaizeRouter			ARCHER[d]			FGR 1.1			BoxRouter 2.0		
	wlen[a]	max.o[b]	ovfl[c]	wlen	max.o	ovfl	wlen	max.o	ovfl	wlen	ovfl	wlen	max.o	ovfl	wlen	max.o	ovfl
adaptec1	104.05	0	0	90.92	2	60	99.61	0	0	113.80	0	88.39	0	0	92.04	0	0
adaptec2	102.97	0	0	92.19	50	2	98.12	0	0	112.56	0	89.89	0	0	94.28	0	0
adaptec3	235.87	0	0	203.44	0	0	214.08	0	0	244.08	0	199.60	0	0	207.41	0	0
adaptec4	211.95	0	0	186.31	0	0	194.38	0	0	221.57	0	179.36	0	0	186.42	0	0
adaptec5	298.08	0	0	264.58	2	2.5K	305.32	2	2	334.09	0	259.86	0	0	270.41	0	0
newblue1	101.83	2	400	92.89	4	2.7K	101.74	16	1.3K	116.08	682	94.27	2	452	92.94	2	394
newblue2	155.07	0	0	136.08	0	0	139.66	0	0	166.50	0	129.40	0	0	134.64	0	0
newblue3	195.51	1.1K	39K	168.42	636	54K	184.4	1.1K	33K	198.77	33K	173.82	374	39K	172.44	364	39K

[a]wirelength: each via is counted as three units of wirelength.
[b]maximum number of overflows on any edge.
[c]total number of overflows.
[d]maximum number of overflows are not reported in [Ozdal and Wong 2007].

Table IV. Comparison Between Published Global Routers and BoxRouter 2.0 on ISPD98 Benchmarks

	Chi Dispersion[a]			BoxRouter1.0			FastRoute2.0[a]			ARCHER[a]			FGR 1.0[a]			BoxRouter2.0[b]			BoxRouter2.0[c]		
name	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu
ibm01	66K	189	15.1	66K	102	8.3	68K	31	0.94	64K	0	11	63K	0	13.8	66529	0	3.5	62659	0	32.8
ibm02	179K	64	47.9	179K	33	34.1	179K	0	1.16	172K	0	25	169K	0	17.7	180053	0	4.6	171110	0	35.9
ibm03	152K	10	35.2	151K	0	16.9	150K	0	0.75	147K	0	10	146K	0	6.5	151185	0	3.5	146634	0	17.6
ibm04	173K	465	54.1	173K	309	23.9	175K	64	1.88	170K	0	24	167K	0	38.5	176765	0	27.4	167275	0	115.9
ibm06	289K	35	80.1	282K	0	33.0	285K	0	2.35	279K	0	23	278K	0	23.8	288420	0	8.4	277913	0	47.4
ibm07	379K	309	122.2	379K	53	50.9	375K	0	2.00	370K	0	25	366K	0	26.1	377072	0	14.4	365790	0	85.9
ibm08	415K	74	113.8	415K	0	93.2	412K	0	2.95	405K	0	42	405K	0	24.7	418285	0	17.1	405634	0	90.1
ibm09	426K	52	125.1	419K	0	63.9	425K	3	2.40	414K	0	37	413K	0	25.6	431298	0	17.1	413862	0	273.1
ibm10	600K	51	212.9	593K	0	95.1	596K	0	3.49	584K	0	45	579K	0	119.9	610680	0	17.2	590141	0	352.4
total	2.7M	1249	806	2.7M	555	20	2.7M	98	17.8	2.6M	0	242	2.6M	0	297	2.7M	0	113	2.6M	0	1151
ratio	1.01	12.7	45.0	1.00	5.7	1.1	1.00	1.0	1.0	0.98	0.0	13.6	0.98	0.0	16.7	1.01	**0.0**	6.3	0.98	**0.0**	58.7

*reference for each router: *Chi* Dispersion [Hadsell and Madden 2003], BoxRouter 1.0 [Cho and Pan 2006], and FastRoute 2.0 [Pan and Chu 2007].
[a]the numbers are quoted from [Pan and Chu 2007], [Roy and Markov 2007], and [Ozdal and Wong 2007] respectively, and runtimes are scaled based on *Chi* Dispersion and BoxRouter 1.0 speed.
[b]tuned for the best runtime.
[c]tuned for the best quality.

Table V. Comparison Between ISPD98 and Our New ISPD98H/I Benchmarks

				ISPD98			ISPD98H			ISPD98I		
name	nets	grids	lb.wlen[b]	v.cap	h.cap.	t.cap	v.cap	h.cap.	t.cap	v.cap	h.cap.	t.cap
ibm01	11507	64 × 64	60142	12	14	26	11	13	24	10	13	23
ibm02	18429	80 × 64	165863	22	34	56	18	29	47	17	29	46
ibm03	21621	80 × 64	145678	20	30	50	17	27	44	17	26	43
ibm04	26163	96 × 64	162734	20	23	43	19	23	42	19	22	41
ibm05	27777	128 × 64	409709	42	63	105	24	44	68	23	44	67
ibm06	33354	128 × 64	275868	20	33	53	16	29	45	16	28	44
ibm07	44394	192 × 64	363537	21	36	57	18	32	50	17	32	49
ibm08	47944	192 × 64	402412	21	32	53	17	28	45	17	27	44
ibm09	50393	256 × 64	411260	14	28	42	11	25	36	11	24	35
ibm10	64227	256 × 64	574407	27	40	67	20	32	52	19	32	51
			total	219	333	552	171	282	453	166	277	443
			ratio	1.00	1.00	1.00	0.78	0.85	0.82	0.76	0.83	0.80

be preferred to runtime in global routing, unless the main purpose of a global router is the integration with placement [Pan and Chu 2006].

6.3 New ISPD98 Benchmarks

As shown in Table IV, BoxRouter 2.0 conquers ISPD98 benchmarks. Therefore, ISPD98 benchmarks are not enough to push the limit of BoxRouter 2.0. The new ISPD07 routing benchmarks in Table II is too time/memory-consuming to perform in-depth study of global routers and to provide insight on algorithm in reasonable turn-around time. Therefore, we choose to reduce the capacities of ISPD98 benchmarks (see Table V) to test the limit of routing research, and try BoxRouter 2.0 on the new sets of benchmarks. To avoid any confusion, we name the two modified ISPD98 benchmarks as follows:

—*ISPD98H(ard) Benchmarks*. with fewer capacities than ISPD98 benchmarks, and which can be marginally completed by BoxRouter 2.0.

—*ISPD98I(mpossible)*[3] *Benchmarks*. with one fewer capacity than *ISPD98H benchmarks*.

Table VI reports the routing results of BoxRouter 1.0, FGR 1.0, and BoxRouter 2.0 on ISPD98H and ISPD98I. Note that BoxRouter 1.0 binary is available and can be downloaded from [UTDA]. For this experiment, we simply use the *default* parameters for BoxRouter 1.0, FGR 1.0, and BoxRouter 2.0 for all the circuits. It shows BoxRouter 2.0 and FGR 1.0 can complete all the circuits in ISPD98H benchmarks which has on average 18% fewer total capacities than original ISPD98.

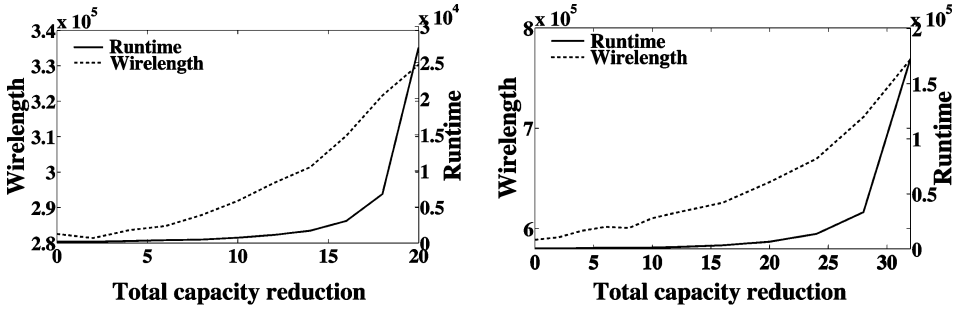
As mentioned in Section 5, congestion-aware Steiner tree is mostly useful for hard cases, as the effect of congestion-awareness can be washed out by global router for easy cases. Thus, we apply node shifting to ISPD98I benchmarks to observe the effect of node shifting clearly, and the results are presented in the last six columns of Table VI. When node shifting is used with edge shifting, the

[3]This only implies that BoxRouter 2.0 cannot complete ISPD98I, and does not claim ISPD98I proven to be unroutable.

Table VI. Comparison Between BoxRouter 1.0/FGR 1.0and BoxRouter 2.0 on New ISPD98H/I Benchmarks

	ISPD98H								ISPD98I							
	BoxRouter1.0			FGR 1.0			BoxRouter2.0			BoxRouter1.0			BoxRouter2.0[a]			BoxRouter2.0[b]	
name	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu	wlen	ovfl	cpu
ibm01	67K	473	8.2	66K	0	139.2	66554	0	122.3	68K	837	9.1	69796	118	852.9	70214	98	892.6
ibm02	189K	4789	38.0	187K	0	1109.0	192827	0	995.0	189K	5770	40.7	202042	40	3.2K	201150	28	3.8K
ibm03	160K	1641	20.8	155K	0	651.0	160182	0	988.2	161K	2214	22.2	166330	58	2.8K	166083	58	3.1K
ibm04	174K	660	22.9	171K	0	236.0	172104	0	386.9	174K	900	23.9	172956	78	2.2K	173669	80	2.4K
ibm05	456K	3953	64.3	435K	0	3.0K	439496	0	4.5K	460K	6155	66.3	460231	212	21.2K	460598	174	26.2K
ibm06	304K	6063	42.2	303K	0	3.0K	308636	0	4.8K	305K	8752	45.0	320264	58	9.2K	323493	16	9.3K
ibm07	389K	2372	59.3	387K	0	2.8K	392795	0	10K	390K	3558	63.2	411960	182	172.9K	410908	180	101.2K
ibm08	438K	4298	96.1	431K	0	2.6K	439338	0	51K	442K	6181	100.8	446468	158	52.3K	447835	110	49.8K
ibm09	445K	3532	71.7	440K	0	4.7K	461898	0	69K	447K	4855	74.7	501790	62	107.2K	505248	52	123.7K
ibm10	635K	16.0K	127.6	681K	0	34K	728956	0	47K	639K	18.5K	129.7	773957	98	239.0K	771019	80	204.7K

[a]the initial congestion-aware Steiner tree is built by only edge shifting [Pan and Chu 2006].
[b]the initial congestion-aware Steiner tree is built by both edge shifting and node shifting in Section 4.1.



(a) ibm02 (b) ibm10

Fig. 12. Runtime exponentially depends on total routing capacity, while wirelength shows quadratic dependency.

number of overflows can be reduced by 17.7% (from 1064 to 876) with negligible overhead in wirelength, compared with edge shifting only.

Routing the harder cases involves significant runtime and wirelength overhead as shown in Table VI. We examine how runtime increases with fewer capacities (difficulty of a circuit) for ibm02 and ibm10 where the most amount of capacity reduction is achieved. Figure 12 shows that the runtime is exponentially dependent on the total capacity. This has an important message to global routing in real practice. If routing capacity estimation considering prerouted nets as well as blockage porosity, is too conservative (substantially fewer capacities than the actual capacities), it may incur unnecessary but significant runtime overhead. Of course, the other way incurs runtime overhead at detailed routing stage.

7. CONCLUSION

Modern VLSI design becomes more complex and denser due to the demand for high-performance and various functionalities, making routability even more challenging. In order to cope with the routability issue, a key to successful design, we propose a new global router, BoxRouter 2.0, which can effectively eliminate congestion. Experiments demonstrate the performance of BoxRouter 2.0 in terms of routability and wirelength/via on ISPD07 and ISPD98 benchmarks. We plan to improve BoxRouter 2.0 continuously, since there are likely additional room for improvement, as indicated by other global routers.

REFERENCES

AHN, K. AND SAHNI, S. 1993. Constrained via minimization. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 12*, 2, 273–282.

ALBRECHT, C. 2001. Global routing by new approximation algorithms for multicommodity flow. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 20*, 5, 622–632.

ALPERT, C. J., GANDHAM, G., HRKIC, M., HU, J., QUAY, S. T., AND SZE, C. 2004. Porosity aware buffered steiner tree construction. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 23*, 4, 517–526.

BRENNER, U. AND ROHE, A. 2003. An effective congestion-driven placement framework. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 22*, 4, 387–394.

BURSTEIN, M. AND PELAVIN, R. 1983. Hierarchical wire routing. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 2*, 4 (Oct), 223–234.

CAO, Z., JING, T., XIONG, J., HU, Y., HE, L., AND HONG, X. 2007. DpRouter: A fast and accurate dynamic-pattern-based global routing algorithm. In *Proceedings of the Asia and South Pacific Design Automation Conference.*

CHANG, C.-C. AND CONG, J. 1999. An efficient approach to multilayer layer assignment with an application to via minimization. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 18*, 5, 608–620.

CHANG, K. C. AND DU, H. C. 1988. Layer assignment problem for three-layer routing. *IEEE Trans. Comput. 37*, 5, 625–632.

CHO, M. AND PAN, D. Z. 2006. BoxRouter: A new global router based on box expansion and progressive ILP. In *Proceedings of the Design Automation Conference.*

CHO, M. AND PAN, D. Z. 2007. BoxRouter: A new global router based on box expansion and progressive ILP. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 12, 2130–2143.

CHO, M., XIANG, H., PURI, R., AND PAN, D. Z. 2006. Wire density driven global routing for CMP variation and timing. In *Proceedings of the International Conference on Computer Aided Design*.

CHU, C. C. N. 2004. FLUTE: Fast lookup table based wirelength estimation technique. In *Proceedings of the International Conference on Computer Aided Design*.

CIESIELSKI, M. J. AND KINNEN, E. 1981. An optimum layer assignment for routing in ICs and PCBs. In *Proceedings of the Design Automation Conference.*

CONG, J. 1997. Challenges and opportunities for design innovations in nanometer technologies. SRC Design Science Concept Papers.

CONG, J., FANG, J., XIE, M., AND ZHANG, Y. 2005. MARS—A multilevel full-chip gridless routing system. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 24*, 3, 382–394.

GAO, J.-R., WU, P.-C., AND WANG, T.-C. 2008. High-performance routing at the nanometer scale. In *Proceedings of the Asia and South Pacific Design Automation Conference.*

HADSELL, R. T. AND MADDEN, P. H. 2003. Improved global routing through congestion estimation. In *Proceedings of the Design Automation Conference.*

HENTSCHKE, R., NARASIMHAN, J., JOHANN, M., AND REIS, R. 2007. Maze routing steiner trees with effective critical sink optimization. In *Proceedings of the International Symposium on Physical Design*.

HTTP://WWW.ISPD.CC/ISPD07_CONTEST.HTML.

HU, J. AND SAPATNEKAR, S. 2000. A timing-constrained algorithm for simultaneous global routing of multimple nets. In *Proceedings of the International Conference on Computer Aided Design*.

HU, J. AND SAPATNEKAR, S. 2002. A survey on multi-net global routing for integrated circuits. *Integration, VLSI J. 31*, 1, 1–49.

JHANG, K. 2000. Minimum crosstalk layer assignment in a three layer HVH channel routing based on linear pseudo Boolean optimization. In *Proceedings of the International Conference on Microelectronics*.

KASTNER, R., BOZORGZADEH, E., AND SARRAFZADEH, M. 2001. An exact algorithm for coupling-free routing. In *Proceedings of the International Symposium on Physical Design*.

KASTNER, R., BOZORGZADEH, E., AND SARRAFZADEH, M. 2002. Pattern routing: Use and theory for increasing predictability and avoiding coupling. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 21*, 7, 777–790.

KAY, R. AND RUTENBAR, R. A. 2000. Wire packing: A strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution. In *Proceedings of the International Symposium on Physical Design*.

KUTZSCHEBAUCH, T. AND STOK, L. 2001. Congestion aware layout driven logic synthesis. In *Proceedings of the International Conference on Computer Aided Design*.

LEE, K.-Y. AND WANG, T.-C. 2006. Post-routing redundant via insertion for yield/reliability improvement. In *Proceedings of the Asia and South Pacific Design Automation Conference.*

MCMURCHIE, L. AND EBELING, C. 1995. PathFinder: A negotiation-based performance-driven router for FPGAs. In *Proceedings of the ACM Symposium on FPGAs*.

NACLERIO, N., MASUDA, S., AND NAKAJIMA, K. 1989. The via minimization problem is NP-complete. *IEEE Trans. Comput. 38*, 11, 1604–1608.

OZDAL, M. M. AND WONG, M. D. F. 2007. ARCHER: A history-driven global routing algorithm. In *Proceedings of the International Conference on Computer Aided Design*.

PAN, M. AND CHU, C. 2006. FastRoute: A step to integrate global routing into placement. In *Proceedings of the International Conference on Computer Aided Design*.

PAN, M. AND CHU, C. 2007. Fastroute 2.0: A high-quality and efficient global router. In *Proceedings of the Asia and South Pacific Design Automation Conference*.

ROY, J. A. AND MARKOV, I. L. 2007. High-performance routing at the nanometer scale. In *Proceedings of the International Conference on Computer Aided Design*.

SEMICONDUCTOR INDUSTRY ASSOCIATION. 2007. *International Technology Roadmap for Semiconductors*. Semiconductor Industry Association.

SHI, C.-J., VANNELLI, A., AND VLACH, J. 1997. Performance-driven layer assignment by integer linear programming and path-constrained hypergraph partitioning. *J. Heurist. 3*, 3, 225–243.

UTDA. *http://www.cerc.utexas.edu/utda*.

WENTING, H., HONG, Y., XIANLONG, H., WEIMIN, C. Y. W., GU, G. J., AND KAO, W. 2001. A new congestion-driven placement algorithm based on cell inflation. In *Proceedings of the Asia and South Pacific Design Automation Conference*.

WESTRA, J., BARTELS, C., AND GROENEVELD, P. 2004. Probabilistic congestion prediction. In *Proceedings of the International Symposium on Physical Design*.

WESTRA, J., BARTELS, C., AND GROENEVELD, P. 2005. Is probabilistic congestion estimation worthwhile? In *Proceedings of the System-Level Interconnect on Prediction*.

WESTRA, J., GROENEVELD, P., YAN, T., AND MADDEN, P. H. 2005. Global routing: Metrics, benchmarks, and tools. In *Proceedings of the IEEE Design Automation Technical Conference on Electronic Design Process*.

WU, D., HU, J., AND MAHAPATRA, R. 2005. Coupling aware timing optimization and antenna avoidance in layer assignment. In *Proceedings of the International Symposium on Physical Design*.

WU, D., HU, J., AND MAHAPATRA, R. 2006. Antenna avoidance in layer assignment. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 25*, 4, 734–74.

WU, D., HU, J., MAHAPATRA, R., AND ZHAO, M. 2004. Layer assignment for crosstalk risk minimization. In *Proceedings of the Asia and South Pacific Design Automation Conference*.

XU, G. R., TIAN, D. Z. P., AND WONG, M. D. F. 2005. CMP aware shuttle mask floorplanning. In *Proceedings of the Asia and South Pacific Design Automation Conference*.

Received July 2007; revised March 2008, September 2008; accepted November 2008