BoxRouter 2.0: Architecture and Implementation of a Hybrid and Robust Global Router

Minsik Cho, Katrina Lu, Kun Yuan, and David Z. Pan
Dept. of ECE, The University of Texas at Austin, Austin, TX 78712
{thyeros, yiotse, kyuan}@cerc.utexas.edu, dpan@ece.utexas.edu

***Abstract*— In this paper, we present BoxRouter 2.0, a hybrid and robust global router with discussion on 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 BoxRouter 2.0 which has strong ability to improve routability and minimize the number of vias with blockages, while minimizing wirelength. BoxRouter 2.0 is improved over [1], but can perform multi-layer routing with 2D global routing and layer assignment. Our 2D global routing is equipped with two ideas: robust negotiation-based A* search for routing stability, and topology-aware wire ripup for flexibility. 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) ISPD98 benchmark for the first time in the literature with the shortest wirelength.**

I. 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 [2]–[5]. 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 limiting routing area and thus make wiring gates more challenging. Therefore, routability should be one of the most critical design objectives in VLSI physical design [6]–[8].

Routability can be enhanced in multiple stages in physical design [9]–[11], 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 many design rules, global routing should eliminate congestion by migrating wires from more to less congested regions with the minimized overhead in wirelength and via.

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

The significance of routability in VLSI global routing has led to many global routing algorithms. Burstein et al. [12] proposed a hierarchical approach to speed up integer programming formulation for global routing, and Kastner [13] proposed a pattern-based global routing. Hadsell et al. [14] presented the *Chi* dispersion router based on linear cost function as well as predicted congestion map, and showed better results than [13]. The multicommodity flow-based global router by Albrecht [15] showed good results and was used in industry, but at the expense of computational effort. After BoxRouter [1] sparked the renewed interest in routing research with significantly improved performance, FastRoute [16], [17] and DpRouter [18] achieved high quality solution 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 of the important layer assignment.

Layer assignment plays critical role for routability, timing, crosstalk, and manufacturability/yield. If excessive number of wires are assigned to a particular layer, it will aggravate congestion and crosstalk [19], [20]. 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 [21]. Length of antenna can be also reduced by layer assignment [22]. Large number of vias due to poor layer assignment can cause routability/pin access problem, as via (even extended via) needs larger area as well as wider spacing than wire. Especially, via minimization becomes more important for nanometer design, as via failure is one of critical manufacturability issues [23], [24].

Recent global routing contest in ISPD-2007 [25] 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.

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

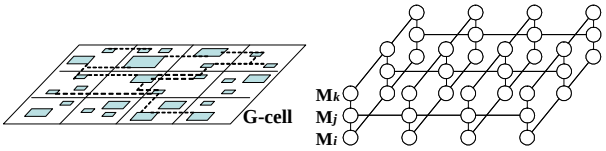
- We propose simple, yet essential dynamic scaling for robust negotiation-based A* search. This prevents a router from spinning out of control by balancing historic cost and present congestion cost, and ensures consistent routability improvement over iterations.
- Instead of ripping up the entire net crossing the congested regions, we perform topology-aware wire ripup which rips up some wires in the congested regions without changing the net topology.
- We propose an integer linear programming (ILP) for via/blockage-aware layer assignment to handle blockages and guarantee the feasibility. Also, we apply progressive ILP technique for via/blockage-aware layer assignment to enhance runtime.
- We achieve a complete routable solution of ISPD98 benchmark in the shortest wirelength for the first time, compared with all published academic global routers. Also, ours finishes the most of number of circuits with comparable wirelength on ISPD07 global routing benchmark, compared with all winning global routers.

The rest of the paper is organized as follows. Section II presents preliminaries. Section III provides an overview of BoxRouter 2.0. Details on our 2D global routing is described in Section IV, then layer assignment is proposed in Section V. Experimental results are discussed in Section VI, followed by conclusion in Section VII.

II. Preliminaries

A. Global Routing Background

The global routing problem can be modeled as a grid graph, where each rectangular region of the circuit can be represented by the same number of vertices as the number of metal layers in the given manufacturing process. Fig. 1 shows a grid graph for routing from a circuit in multi-metal layer manufacturing process. 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 [8], [13]. Thus, a global routing is to find paths that connect the pins inside the G-cells through the graph for every net with minimum number of overflows [8]. Since a net may have complex topology, it can be decomposed into two pin *wires* with Rectilinear Minimum Steiner Tree [1], [16].



(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 [1].

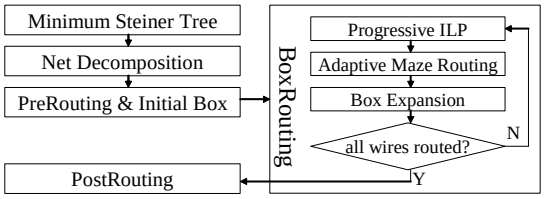


Fig. 2. BoxRouter 1.0 overall flow [1]

B. BoxRouter 1.0

BoxRouter 1.0 [1] 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, BoxRouter 1.0 performs progressive integer linear programming (ILP) and adaptive maze routing to effectively diffuse the congestion as in Fig. 2. 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 [1] shows significantly superior results on ISPD98 benchmark, compared with [13]–[15].

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 simple L-shape pattern [26], [27]) does not hold. In detail, its progressive ILP formulation for routing only considers 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.

C. Negotiation-based Routing

It is shown that negotiation-based routing is effective in congestion elimination for FPGA [28]. The key idea of 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 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 routing cost will guide a router to avoid the presently congested edges as well as previously congested edges. This is very efficient technique to spread out wires to less congested regions.

III. Overview of BoxRouter 2.0

In this section, we give the overview of BoxRouter 2.0 shown in Fig. 3. The early steps of BoxRouter 2.0 are inspired by BoxRouter [1], but ours is radically different in a sense that we have more powerful and systematic way of removing congestion and assigning layers to wires. BoxRouter 2.0 has two major steps, 2D global routing (Section IV) and layer

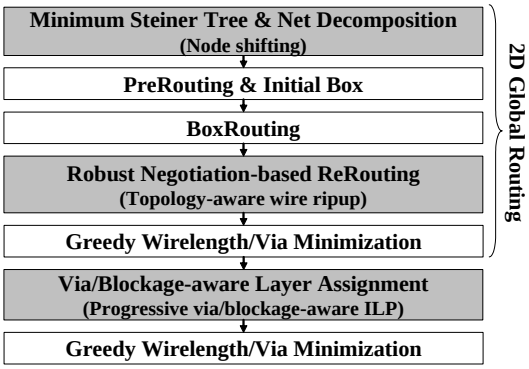
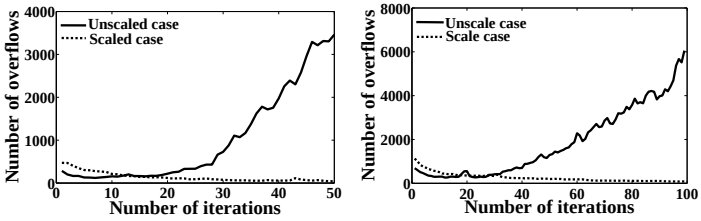


Fig. 3. The overview of BoxRouter 2.0



(a) ibm01 (b) ibm04

Fig. 4. Dynamically scaled A* search reduces congestions robustly and stably over iterations.

assignment (Section V). 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 for every wire is fixed as a constant.

IV. 2D Global Routing

In this section, we present 2D global routing algorithm. As BoxRouter 2.0 is inspired by PreRouting and BoxRouting of BoxRouter [1], we take them to generate the initial routing solution as in Fig. 2. However, we improve routability considerably by our negotiation-based ReRouting. Our technical contributions in 2D global routing include the following:

1) **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 IV-A.
2) **Topology-aware wire ripup:** The goal is to enhance solution quality further by providing more flexibility in rerouting, while honoring the current routing topology. This is presented in Section IV-B.

A. Robust Negotiation-based A* search

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

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

where regarding an edge e, $h^i(e)$ is a historic cost at i-th iteration, $p(e)$ is the present congestion cost (utilization), and $d(e)$ is the distance from e to the target.

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 over $p(e)$. 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 continuous improvement in routability, the balance between two costs has to be kept.

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

$$\alpha = \frac{max_e[h^i(e)]}{p(e)|_{1.0}} \tag{2}$$

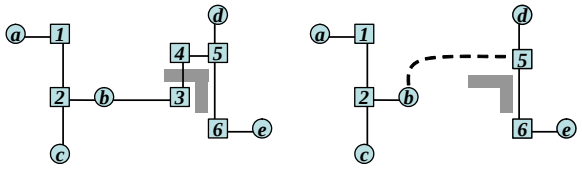
where $p(e)|_{1.0}$ indicates the congestion cost when there is no available routing capacity in an edge e. Insight behind such α is to make a presently congested edge (no more routing capacity available) passing as expensive as a previously congested edge passing. This will discourage creating new overflows, while avoiding previously congested edges.

Fig. 4 shows the effect of robust negotiation-based A* search by comparing the scaled case (Eq. (2)) and unscaled case (α=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. With larger α, we may delay spinning out of control, but it will eventually occur after larger number of iterations. 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 large number of iterations.

B. Topology-aware Wire Ripup

When a wire is selected for rerouting, we explore larger flexibility by ripping up some adjacent wires in the same net, while honoring the current routing topology. The reason we need to honor the current topology is because an abrupt change in congestion map can misguide a router with inaccurate congestion estimation.

Consider the example in Fig. 5 where pins are in circle (a, b, c, d, e) and Steiner points or bends are in square $(1, 2, 3, 4, 5, 6)$. As wire 3-4 in Fig. 5 (a) is passing through a congested region in dark area, it will be ripped up for rerouting. Moreover, two connected wires, wire b-3 and 4-5 are ripped up together as shown in Fig. 5 (b). The motivation behind our ripping up is that a Steiner point or bend (which



(a) before wire ripup (b) after wire ripup

Fig. 5. Topology-aware wire ripup improves routing flexibility by ripping up some connected wires, but honors the current routing topology.

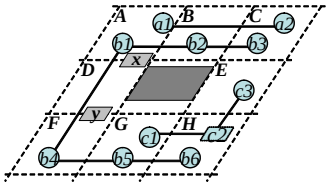
is not a pin) with degree two such as 3 and 4 are not critical in terms of routing topology, as they simply bridge two wires. Thus, ripping up wire b-3-4-5 provides more flexibility in terms of rerouting, while honoring the current topology.

V. Layer Assignment

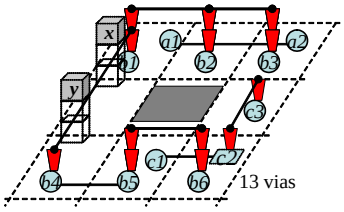
In this section, we propose a layer assignment for via-minimization based on progressive integer linear programming (ILP). When 2D global routing is finished, layer assignment follows to distribute the wires across the layers. Layer assignment impacts several design objectives, such as timing, noise, and manufacturability, but our layer assignment mainly focuses on via minimization without altering routing topology. This problem similar to constrained via minimization (CVM) [29]–[31] which is NP-complete [32].

A. Via/Blockage-aware Layer Assignment

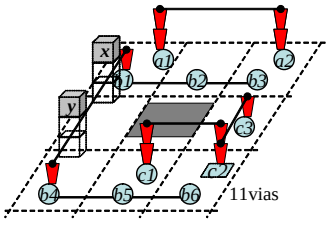
Depending on layer assignment, the number of vias can be significantly different. Fig. 6 shows an example of layer assignment for via minimization, where net a, b, and c are routed through 2D global routing cells, and pins are shown in circle, while a Steiner point ($c2$) in square. The example assumes four metal layers ($M1$-$M4$), where $M1$ and $M3$ are for horizontal wires, $M2$ and $M4$ are for vertical wires, and all the pins on $M1$. Further, a single routing capacity is assumed



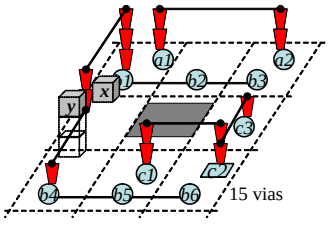
(a) 2D global routing result for net a,b, and c with blockages when $M1$-$M4$ are superposed



(b) suboptimal via-aware layer assignment of 13 vias with blockage x and y on $M4$



(c) optimal via-aware layer assignment of 11 vias with blockage x and y on $M4$



(d) optimal via-aware layer assignment of 15 vias with blockage x on $M2$ but y on $M4$

Fig. 6. 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).

TABLE I

The notations in Fig 7.

$W(i,s)$	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 $s \in P(i)$

$$
\begin{aligned}
\textbf{min :}\quad & \textstyle\sum_i \sum_{s\in P(i)} (T_{is} - B_{is}) - \alpha \sum_{i,j,k} z_{ijk} && (\alpha \gg 1)\\
\textbf{s.t :}\quad & z_{ijk} \in \{0,1\} && \forall i,j,k\\
& \textstyle\sum_k z_{ijk} \le 1 && \forall i,j,k\\
& \textstyle\sum_k k \cdot z_{ijk} = l_{ij} && \forall i,j,k\\
& B_{is} \le l_{ij} \le T_{is} && \forall (i,j) \in W(i,s)\\
& B_{is} = M1 && \forall s \in N(i)\\
& \textstyle\sum_{(i,j,k)\in C(e)} z_{ijk} \le r_e && \forall e
\end{aligned}
$$

Fig. 7. ILP formulation for via/blockage-aware layer assignment

for each edge. If a greedy approach (a shorter net is assigned to lower layer) is adopted, it will result in Fig. 6 (b) with 13 vias. But, Fig. 6 (b) has 2 more vias (18%) than the optimal assignment in Fig. 6 (c). This is simply because the greedy approach cannot capture the global view.

Also, as 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 Fig. 6 (c) and Fig. 6 (d) where the blockage x is located in $M4$ and $M2$, respectively. In Fig. 6 (c), both x and y are on $M4$, enabling to route wire $b1-b4$ on $M2$. However, in Fig. 6 (d), wire $b1-b4$ cannot be routed *as it is*, as x is on $M2$ while y is on $M4$. Wire $b1-b4$ should be chopped into two pieces such that it can shuttle from $M2$ to $M4$ as in Fig. 6 (d). This issue can be easily addressed by chopping wires, wherever a blockage exits, but this may result in not only unnecessary vias, but also computational inefficiency due to more object to handle.

Motivated by the idea in [1], we propose a ILP formulation for via/blockage-aware layer assignment as shown in Fig. 7, where the objective is to complete as many wires as possible, while minimizing the number of vias. This formulation is feasible for any blockage distribution. The unassigned wires after solving ILP will be picked up by a maze routing like [1] , but ours is simpler and faster (it only needs to shuttle between layers). Therefore, less number of wires will be chopped than the approach of chopping wires for each blockage, resulting in less number of vias in shorter runtime.

B. Progressive ILP for Via/Blockage-aware Layer Assignment

ILP is computationally expensive, as most solvers use branch-and-bound algorithm. Thus, to apply ILP to industrial designs, the problem size should be tractable, while maintaining the global view. We adapt the idea of box expansion and progressive ILP [1] for our layer assignment.

TABLE II

ISPD07 IBM BENCHMARKS [25].

name[a]	nets	grids	v.cap[b]	h.cap[b]	placer
adaptec1	219794	324x324	70	70	Capo
adaptec2	260159	424x424	80	80	mPL6
adaptec3	466295	774x779	62	62	Dragon
adaptec4	515304	774x779	62	62	APlace3
adaptec5	867441	465x468	110	110	mFAR
newblue1	331663	399x399	62	62	NTUplace 3.0
newblue2	463213	557x463	110	110	FastPlace 3.0
newblue3	551667	973x1256	80	80	Kraftwerk

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

TABLE III

ISPD98 IBM BENCHMARKS [34].

name	nets	grids	v.cap	h.cap	t.cap [a]	lb.wlen [b]
ibm01	11507	64x64	12	14	26	60142
ibm02	18429	80x64	22	34	56	165863
ibm03	21621	80x64	20	30	50	145678
ibm04	26163	96x64	20	23	43	162734
ibm05	27777	128x64	42	63	105	409709
ibm06	33354	128x64	20	33	53	275868
ibm07	44394	192x64	21	36	57	363537
ibm08	47944	192x64	21	32	53	402412
ibm09	50393	256x64	14	28	42	411260
ibm10	64227	256x64	27	40	67	574407

[a] total capacity: v.cap + h.cap
[b] lower bound wlen computed by GeoSteiner 3.1 [35]

VI. 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 [16] based on Flute [33] is adopted. We use ISPD07 benchmark to demonstrate BoxRouter 2.0 Also, we apply BoxRouter 2.0 to ISPD98 benchmark as well. Details on ISPD07 and ISPD98 benchmark are presented in Table II and III respectively.

A. ISPD07 Benchmark

We report the results of other global routers entered ISPD-2007 routing contest [25] as well as that of BoxRouter 2.0 on ISPD07 benchmark in Table IV. Regarding wirelength, ours is comparable with FGR, but significantly better than BoxRouter 1.9, MaizeRouter (especially for 3D benchmark). However, BoxRouter 2.0 completes the most number of circuits (12 out of 16), which ties with BoxRouter 1.9, but with less number of total/maximum overflows. For the uncompleted circuits (newblue1 and newblue3), we have smaller number of maximum overflows, which may be easily fixed during detailed routing. All the results prove that BoxRouter 2.0 has strong routability, which is the utmost goal of global routing, and provides high quality solution in terms of wirelength/via. BoxRouter 2.0 requires 1.5GB memory and takes more than 2 days for the biggest newblue3.3d.

B. ISPD98 Benchmark

We use ISPD98 benchmark to compare BoxRouter 2.0 with recently published global routers, Labyrinth, *Chi* Dispersion, DpRouter, BoxRouter 1.0, and FastRoute 2.0. Note that as the binaries of ISPD07 contestants are not available, we cannot compare with them on ISPD98 Benchmark. Table V shows the performance of each router on ISPD98 benchmark. 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 is the only one which completes ISPD98 benchmark without any overflow. We tune BoxRouter 2.0 for runtime and quality respectively, and compare both results with other global routers as shown in Table V. When tuned for runtime, although slower than FastRoute 2.0 or DpRouter, ours is 4-12x faster than the others. But, better congestion distribution (no overflow) than FastRoute 2.0 and DpRouter will be significantly rewarded in detailed routing by huge speed-up. Therefore, higher quality solution should be preferred to runtime in global routing, unless the main purpose of global router is the integration with placement [16]. When tuned for quality, ours achieves the best wirelength.

VII. 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 routability issue, we propose BoxRouter 2.0 which can robustly 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 in terms of quality and runtime.

References

[1] M. Cho and D. Z. Pan, "BoxRouter: A New Global Router Based on Box Expansion and Progressive ILP," in *Proc. Design Automation Conf.*, July 2006.
[2] International Technology Roadmap for Semiconductors (ITRS), 2006.
[3] J. Cong, "Challenges and opportunities for design innovations in nanometer technologies," in *SRC Design Science Concept Papers*, 1997.
[4] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "An Exact Algorithm for Coupling-Free Routing," in *Proc. Int. Symp. on Physical Design*, Apr 2001.
[5] D. Wu, J. Hu, and R. Mahapatra, "Coupling Aware Timing Optimization and Antenna Avoidance in Layer Assignment," in *Proc. Int. Symp. on Physical Design*, Apr 2005.
[6] J. Hu and S. Sapatnekar, "A Survey On Multi-net Global Routing for Integrated Circuits," *Integration, the VLSI Journal*, vol. 31, no. 1, pp. 1–49, 2002.
[7] ——, "A Timing-Constrained Algorithm for Simultaneous Global Routing of Multiple Nets," in *Proc. Int. Conf. on Computer Aided Design*, Nov 2000.
[8] J. Westra, P. Groeneveld, T. Yan, and P. H. Madden, "Global Routing: Metrics, Benchmarks, and Tools," in *IEEE DATC Electronic Design Process*, Apr 2005.
[9] T. Kutzschebauch and L. Stok, "Congestion aware layout driven logic synthesis," in *Proc. Int. Conf. on Computer Aided Design*, Nov 2001.
[10] H. Wenting, Y. Hong, H. Xianlong, C. Y. W. Weimin, G. J. Gu, and W. Kao, "A new congestion-driven placement algorithm based on cell inflation," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2001.
[11] U. Brenner and A. Rohe, "An Effective Congestion-Driven Placement Framework," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, 2003.
[12] M. Burstein and R. Pelavin, "Hierarchical Wire Routing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 2, no. 4, pp. 223–234, Oct 1983.
[13] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 7, pp. 777 – 790, July 2002.

TABLE IV

COMPARISON BETWEEN ISPD07 CONTESTANTS AND BOXROUTER 2.0 ON ISPD07 BENCHMARKS.

	BoxRouter 1.9 [25]			FGR [25]			MaizeRouter [25]			FastRouter [25]			BoxRouter 2.0		
name	wlen[a]	max.o[b]	ovfl[c]	wlen	max.o	ovfl	wlen	max.o	ovfl	wlen	max.o	ovfl	wlen	max.o	ovfl
adaptec1.2d	58.84	0	0	55.8	0	0	62.26	0	0	90.47	4	122	58.37	0	0
adaptec2.2d	55.69	0	0	53.69	0	0	57.23	0	0	82.46	12	500	55.69	0	0
adaptec3.2d	140.87	0	0	133.34	0	0	137.75	0	0	202.53	0	0	137.96	0	0
adaptec4.2d	128.75	0	0	126.05	0	0	128.45	0	0	170.8	0	0	127.79	0	0
adaptec5.2d	164.32	0	0	155.82	0	0	176.69	2	2	251.68	76	9680	162.11	0	0
newblue1.2d	51.13	2	400	47.51	10	1218	50.93	16	1348	74.1	32	1934	51.13	2	400
newblue2.2d	79.78	0	0	77.67	0	0	79.64	0	0	114.95	0	0	78.68	0	0
newblue3.2d[d]	111.64	1088	38976	108.18	1090	36970	114.63	1236	32588	154.59	1306	34236	111.61	1088	38958
adaptec1.3d	104.05	0	0	90.92	2	60	99.61	0	0	248.95	4	122	92.04	0	0
adaptec2.3d	102.97	0	0	92.19	50	2	98.12	0	0	244.41	12	500	94.28	0	0
adaptec3.3d	235.87	0	0	203.44	0	0	214.08	0	0	523.21	0	0	207.41	0	0
adaptec4.3d	211.95	0	0	186.31	0	0	194.38	0	0	469.34	0	0	186.42	0	0
adaptec5.3d	298.08	0	0	264.58	2	2480	305.32	2	2	707.86	76	9894	270.41	0	0
newblue1.3d	101.83	2	400	92.89	4	2668	101.74	16	1348	248.26	34	2602	92.94	2	394
newblue2.3d	155.07	0	0	136.08	0	0	139.66	0	0	379.6	0	0	134.64	0	0
newblue3.3d[d]	195.51	1088	38976	168.42	636	53648	184.4	1058	32840	442.72	1306	34236	172.44	364	38958

[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] newblue3.2d and newblue3.3d are proven to be unroutable.

TABLE V

COMPARISON BETWEEN PUBLISHED GLOBAL ROUTERS AND OURS ON ISPD98 BENCHMARK

	Labyrinth [13]			*Chi* Dispersion [14]			DpRouter[a][18]			BoxRouter 1.0 [1]			FastRoute 2.0[a][17]			BoxRouter 2.0(r)[b]			BoxRouter 2.0(q)[c]		
name	wlen	ovfl	cpu(s)	wlen	ovfl	cpu(s)	wlen	ovfl	cpu(s)	wlen	ovfl	cpu(s)	wlen	ovfl	cpu(s)	wlen	ovfl	cpu(s)	wlen	ovfl	cpu(s)
ibm01	77K	398	21.2	66006	189	15.1	63857	125	0.51	65588	102	8.3	68489	31	0.94	66529	0	3.5	62659	0	32.8
ibm02	205K	492	34.5	178892	64	47.9	178261	3	1.26	178759	33	34.1	178868	0	1.16	180053	0	4.6	171110	0	35.9
ibm03	185K	209	36.3	152392	10	35.2	150663	0	0.78	151299	0	16.9	150393	0	0.75	151185	0	3.5	146634	0	17.6
ibm04	197K	882	83.5	173241	465	54.1	172608	165	1.93	173289	309	23.9	175037	64	1.88	176765	0	27.4	167275	0	115.9
ibm06	346K	834	104.3	289276	35	80.1	286025	14	2.41	282325	0	33.0	284935	0	2.35	288420	0	8.4	277913	0	47.4
ibm07	449K	697	228.1	378994	309	122.2	379133	99	2.94	378876	53	50.9	375185	0	2.00	377072	0	14.4	365790	0	85.9
ibm08	470K	665	238.7	415285	74	113.8	412308	56	3.34	415025	0	93.2	411703	0	2.95	418285	0	17.1	405634	0	90.1
ibm09	481K	505	359.3	427556	52	125.1	419199	47	2.56	418615	0	63.9	424949	3	2.40	431298	0	17.1	413862	0	273.1
ibm10	680K	588	435.7	599937	51	212.9	598460	46	4.14	593186	0	95.1	595622	0	3.49	610680	0	17.2	590141	0	352.4
total	3089K	5.2K	1541.6	2682K	1249	806.4	2661K	555	19.9	2657K	497	419.3	2665K	98	17.9	2700K	0	113.4	2601K	0	1151.1
ratio	1.16	53.8	86.0	1.01	12.7	45.0	1.00	5.7	1.1	1.00	5.1	23.4	1.00	1.0	1.0	1.01	**0.0**	6.3	0.98	**0.0**	58.7

[a] the numbers are quoted from [18] and [17] respectively, and runtimes are scaled based on Labyrinth speed.
[b] tuned for runtime, [c] tuned for quality

[14] R. T. Hadsell and P. H. Madden, "Improved Global Routing through Congestion Estimation," in *Proc. Design Automation Conf.*, Jun 2003.

[15] C. Albrecht, "Global Routing by New Approximation Algorithms for Multicommodity Flow," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 5, pp. 622–632, May 2001.

[16] M. Pan and C. Chu, "FastRoute: A Step to Integrate Global Routing into Placement," in *Proc. Int. Conf. on Computer Aided Design*, Nov 2006.

[17] ——, "FastRoute 2.0: A High-quality and Efficient Global Router," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2007.

[18] Z. Cao, T. Jing, J. Xiong, Y. Hu, L. He, and X. Hong, "DpRouter: A Fast and Accurate Dynamic-Pattern-Based Global Routing Algorithm," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2007.

[19] R. Kay and R. A. Rutenbar, "Wire packing: A strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution," in *Proc. Int. Symp. on Physical Design*, 2000.

[20] D. Wu, J. Hu, R. Mahapatra, and M. Zhao, "Layer assignment for crosstalk risk minimization," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2004.

[21] M. Cho, H. Xiang, R. Puri, and D. Z. Pan, "Wire Density Driven Global Routing for CMP Variation and Timing," in *Proc. Int. Conf. on Computer Aided Design*, Nov 2006.

[22] D. Wu, J. Hu, and R. Mahapatra, "Antenna avoidance in layer assignment," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, 2006.

[23] G. Xu, L. Huang, D. Z. Pan, and D. F. Wong, "Redundant-Via Enhanced Maze Routing for Yield Improvement," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2005.

[24] K.-Y. Lee and T.-C. Wang, "Post-Routing Redundant Via Insertion for Yield/Reliability Improvement," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan 2006.

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

[26] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic Congestion Prediction," in *Proc. Int. Symp. on Physical Design*, Apr 2004.

[27] ——, "Is Probabilistic Congestion Estimation Worthwhile?" in *Proc. System-Level Interconnect Prediction*, Apr 2005.

[28] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," in *ACM Symposium on FPGAs*, Feb 1995.

[29] K. C. Chang and H. C. Du, "Layer assignment problem for three-layer routing," *IEEE Trans. on Computers*, vol. 37, 1988.

[30] C.-C. Chang and J. Cong, "An efficient approach to multilayer layer assignment with an application to via minimization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, 1999.

[31] K. Ahn and S. Sahni, "Constrained via minimization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, 1993.

[32] N. Naclerio, S. Masuda, and K. Nakajima, "The via minimization problem is NP-complete," *IEEE Trans. on Computers*, vol. 38, 1989.

[33] C. C. N. Chu, "FLUTE: Fast Lookup Table Based Wirelength Estimation Technique," in *Proc. Int. Conf. on Computer Aided Design*, Nov 2004.

[34] http://www.ece.ucsb.edu/∼kastner/labyrinth/.

[35] http://www.diku.dk/geosteiner/.