A Procedure for Improving the Distribution of Congestion in Global Routing

Daohang Shi, Azadeh Davoodi, Jeffrey Linderoth
University of Wisconsin - Madison
1415 Engineering Dr. Madison WI USA 53706
dshi7@wisc.edu

***Abstract*—This work introduces a procedure which takes as input a global routing solution that is already improved for routability based on the traditional total overflow (TOF) metric, and then improves the distribution of congestion without increasing the TOF. Our router is able to significantly decrease the number of edges in undesirable ranges of congestion by optimizing a convex piece-wise linear penalty function. The penalties are flexible and may be specified by the user. In our experiments, using the already-optimized global routing solutions of the ISPD'11 benchmarks—mostly have 0 units of TOF—we show the number of edges which are utilized very close to capacity can be significantly reduced. This work is the first to explicitly target improving the *distribution* of edge congestion corresponding to an already-optimized global routing solution without sacrificing the TOF.**

I. Introduction

Routing in advanced VLSI technology nodes has become a major challenge. A main goal of global routing is to minimize the effort of the detailed routing as much as possible. Modern global routing requires considering factors such as wire sizes of individual metal layers, layer-specific routing blockages, pins assigned to higher metal layers, and local congestion within a global cell . Some recent works considered these factors during global routing [7], [6], [9], [10], [12].

The majority of recent global routing procedures target improving routability which is driven by minimizing a total overflow metric (denoted by TOF) [2], [3], [4], [8], [13], [14], [15]. TOF is sum of edge overflows in the global routing grid-graph. However, minimizing TOF may no longer be sufficient for routability. Thus the authors in [12] propose two rank-based metrics denoted by $ACN(x)$and $WCI(y)$, where $ACN(x)$ measures the average congestion of the top $x\%$ most-congested edges, and $WCI(y)$ measures the percentage of nets with congestion of at least $y\%$. A fast global router is then proposed in [7] with the main purpose to quickly build a congestion map to interact with a routability-driven placer and not to create an improved global routing solution; the impact on TOF is unclear and not shown in the experiments.

In this work we propose a procedure to shape the distribution of an already-optimized global routing solution while guaranteeing that TOF is not increased. Shaping of the distribution is done in a controlled manner, using a convex penalty function; the user is allowed to associate a higher penalty to minimize the edges which will fall within undesirable ranges of congestion.

In our experiments we use as input the ISPD'11 routability-driven benchmarks which are already optimized by the NCTU-GR 2.0 global router [6] and mostly have 0 units of TOF. Alpert et al. note in [1] that in a design that has 0 TOF, it is also desirable to reduce the number of edges which have a congestion ratio close to 1; these are the edges that are utilized close to their capacities and reducing such edges can provide slack and flexibility at the detailed routing stage when additional factors need to be considered. Therefore in our experiments we show the number of edges with congestion ratio above 0.9, i.e. utilized close to capacity, can be reduced, on-average by 12.83% and as high as 23.80% without increase in TOF.

Overall, we show that a significant improvement can be made to the distribution of the congested edges in an already-optimized global routing solution on realistic modern design instances while guaranteeing that the total overflow is not increased.

II. Preliminaries and Basic Notation

We are given a grid-graph $G = (V, E)$ for global routing of dimension $V = X \times Y \times L$, where L represents the number of metal layers. We are also given a set of nets $\mathcal{N}$ to route. Each net is identified by some terminals which are a subset of the vertices in V. In modern designs, some of these terminals may be *virtual* indicating that they may be located at the higher metal layers.

Each edge $e \in E$ has a constant capacity c_e and is associated with an individual (sum of) wire size and spacing denoted by s_e. It may also have a constant blockage amount of b_e for example representing routing resources taken by pre-routed power delivery or clock networks. Intuitively, edge e represents the boundary of two adjacent global cells and its capacity represents the available routing resources on the boundary when ignoring its routing blockage.

We are also given an initial solution where for each net $i \in \mathcal{N}$, we have a Steiner tree $\hat{t}_i$, a subset of the edges in E that connects the terminals of net i. Each edge $e \in E$ has a utilization $\hat{u}_e$, which is the amount of routing resource consumed from edge e by the initial solution. The overflow of an edge in the initial solution is given by $\max(\hat{u}_e - (c_e - b_e), 0)$ and the total overflow denoted by $\hat{TOF}$ is the sum of all the edge overflows. We wish to improve the distribution of this initial solution while ensuring the total overflow of the global routing solution does not exceed its current value $\hat{TOF}$.

We also define a congestion ratio r_e for each edge e which is given by $\hat{r}_e = \frac{\hat{u}_e + b_e}{c_e}$. A congestion ratio less than or equal to 1 corresponds to an edge with 0 overflow. Our routing framework models all the factors captured in the ISPD'11 benchmarks. The reader is referred to [6], [9], [10], [11] for more details about these benchmarks.

III. Routing Framework

Our router works on non-overlapping routing subregions which are identified around the congested edges in the input solution. In this section, we first describe how our routing subregions are created (Section III-A). We then present a mathematical formulation to be solved inside each subregion (Section III-B), followed by our global routing procedure for solving the formulation (Section III-C).

A. Forming the Routing Subregions

We start by visiting the edges in decreasing order of their initial congestion ratios ($\hat{r}_e$ for edge e). When visiting an edge, we first create a 3D subregion of fixed size centered around the edge. The shape of the subregion is a hyper-cube which extends over all the metal layers. (In our experiments we use subregion size of 20x20x9 which we found appropriate based on typical global grid dimension and number of metal layers for ISPD'11 benchmarks [11].)

978-3-9815370-6-2/DATE16/

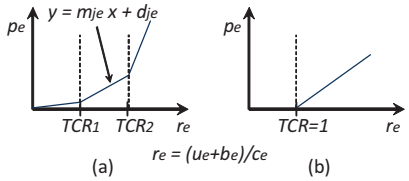

Fig. 1. (a) Mapping edge congestion ratio to edge penalty based on a convex piece-wise linear function; (b) Variation when edge penalty is its overflow.

If the subregion of an edge overlaps with a previous subregion, then the subregion will be dropped so that subregions will be non-overlapping. The process stops as soon as the sum of the volume of the subregions corresponding to the edges visited so far becomes at least 25% of the overall routing space. The remaining edges (which will have a lower congestion ratio) won't be visited.

For each routing subregion $\mathcal{R} = (V_R, E_R)$, we then identify a subset of the nets that are fully or partially routed inside $\mathcal{R}$. If a route crosses the boundary of $\mathcal{R}$, the crossing point will be added as a new terminal of the corresponding net. If a net terminal falls outside $\mathcal{R}$, it will be eliminated from the set of the terminals of the net. In the end, for subregion $\mathcal{R}$, the new set of nets to be routed is denoted by $\mathcal{N}_\mathcal{R}$ and includes those nets that have one or more terminals inside $\mathcal{R}$, and/or have a "pseudo-terminal" on the boundary.

The above procedure creates independent subregions that will be processed in parallel. The routing of each net will be completely inside the subregion and will more effectively utilize the 3D-routing resources. Note that the above procedure ensures the routing solution outside the re-routed subregions remains intact which is useful to ensure minimal change to the already-optimized initial solution.

B. Formulation for Improving the Congestion Distribution

Here we present a mathematical formulation to precisely describe the routing problem that we propose to be solved inside each subregion. We first describe the objective optimized by the formulation, and then describe the formulation as a mixed-integer linear program.

1) Metric for Shaping the Congestion Distribution: The objective of our formulation is to minimize a convex, piecewise-linear penalty function of the edge congestion ratios $r_e = \frac{u_e+b_e}{c_e}$ with u_e, c_e and b_e representing the utilization, capacity and blockage of edge e respectively. The piecewise-linear function is given as a set of slope and y-intercept pairs $J_e = \{(m_{je}, d_{je}) \mid j = 1, 2 \ldots, |J_e|\}$, so that the penalty for edge $e \in E$ is $p_e = m_{je} r_e + d_{je}$ for some $j \in J_e$.

Figure 1(a) depicts a typical example of the penalty function. The x-axis is the congestion ratio r_e, and the y-axis is the penalty p_e. The intervals on the x-axis represent thresholds to specify ranges of interest for congestion ratios which are denoted by (TCRs). Figure 1(a) shows a realization of the penalty function with three line segments. Assume $TCR_1 = 1$ and $TCR_2 = 1.1$. The edges with $r_e \leq 1$ do not have overflow, therefore the associated line segment has a very small slope representing a very small penalty. For the edges with congestion ratio between 1 and 1.1, the line segment has a higher slope indicating a higher penalty. For the remaining range of $r_e > 1.1$, the penalty is incurred at a much higher rate. The objective is to minimize the sum of these edge penalties, which we denote by *PCR* (for Penalized Congestion Ratios).

The piecewise-linear penalty function *PCR* is extremely flexible. For example as shown in Figure 1(b), by creating only the two slope-intercept pairs $J_e = \{(0,0), (1,-1)\}$ $\forall e \in E$, the *PCR* metric becomes equivalent to the total overflow metric TOF (when there is no blockage and if all edge capacities are equal).

2) Mixed-Integer Linear Programming Formulation: Given a routing region $\mathcal{R} = (V_\mathcal{R}, E_\mathcal{R})$ and nets $\mathcal{N}_\mathcal{R}$, we propose a mixed-integer linear programming (MILP) formulation that identifies nets that will improve the PCR metric. We first give the following definitions.

- $\mathcal{T}_i$: set of candidate routes for net $i \in \mathcal{N}_\mathcal{R}$.
- Parameter $a_{te} = 1$ if route $t \in \mathcal{T}_i$ contains edge $e \in E_\mathcal{R}$.
- Piecewise linear functions given as a set of slope-intercept pairs $J_e = \{(m_{je}, d_{je}) \mid j = 1, 2 \ldots |J_e|\}$ $\forall e \in E_\mathcal{R}$.
- Binary variable x_{it} set to 1 if route $t \in \mathcal{T}_i$ is selected for net i and 0 otherwise.
- Real variables $u_e \geq 0$, $o_e \geq 0$. and $p_e \geq 0$ representing the utilization, overflow, and penalty for each edge $e \in E_\mathcal{R}$.

Also recall that parameters c_e, b_e, and s_e represent the capacity, blockage, and sum of wire size and spacing corresponding to edge e as defined in Section II. With the above definitions, the MILP is expressed as follows:

$$\min_{x,p,u,r,o} PCR = \sum_{e \in E_\mathcal{R}} p_e$$

$$\sum_{t \in \mathcal{T}_i} x_{it} = 1 \quad \forall i \in \mathcal{N}_\mathcal{R} \tag{1}$$

$$s_e \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}_i} a_{te} x_{it} = u_e \quad \forall e \in E_\mathcal{R} \tag{2}$$

$$\frac{u_e + b_e}{c_e} = r_e \quad \forall e \in E_\mathcal{R} \tag{3}$$

$$p_e \geq m_{je} r_e + d_{je} \quad \forall e \in E_\mathcal{R} \ \forall j \in J_e \tag{4}$$

$$u_e - c_e + b_e \leq o_e \quad \forall e \in E_\mathcal{R} \tag{5}$$

$$\sum_{\forall e \in E_\mathcal{R}} o_e \leq \hat{TOF} \tag{6}$$

In this formulation, the first set of equations states that for each net in $\mathcal{R}$, only one route should be selected from the set of its routing trees. The second set of equations computes the utilization for each edge in region $\mathcal{R}$. (Here if a route t includes edge e, it contributes s_e units to the utilization of the edge where s_e is a constant parameter representing the sum of wire size and spacing on the layer corresponding to edge e.) The third set of equations computes the congestion ratio r_e for edge e using its utilization u_e and the constant parameters c_e and b_e corresponding to the edge capacity and blockage. The fourth set of inequalities computes the edge penalties p_e using the specified parameters m_{je} and d_{je}. The objective PCR minimizes the sum of the edge penalties. Finally, the fifth and sixth set of inequalities compute the edge overflow o_e and bound the sum of all edge overflows to be at most the total overflow of the initial solution which was denoted by $\hat{TOF}$ (in Section II).

The presented formulation selects a new route for each net from all its possible routing trees inside the subregion such that in the resulting routing solution, the sum of the penalized congestion ratios (*PCR*) is minimized. *The objective of the formulation is designed to "shape" the distribution of the congestion away from highly-congested edges.*

C. Global Routing Procedure

Our global routing procedure solves the above MILP formulation as a sequence of linear programs (LPs) in which only a subset of "critical" net variables are considered. This allows to develop a fast procedure. Furthermore, each LP updates the candidate routes defined by the set $\mathcal{N}_R$ by *adding* new candidate routes to it. We solve a sequence of 20 LPs to create a final set of candidate routes for each net. Finally, the route of a net is found by solving the MILP (mixed-integer) formulation, described in the previous subsection, by setting $\mathcal{N}_R$ to be the set of candidate routes obtained from the last LP.

1) Creating A Smaller-Sized Formulation: After trying various strategies, we follow the following procedure to consider a small number of nets for each LP.

1. For subregion $\mathcal{R}$, sort all the edges $e \in \mathcal{R}$ in descending order of their congestion ratio. (The congestion ratio r_e is directly taken from the initial solution in the first LP, and from the corresponding variable assignment from the previous LP.)
2. For a visited edge e, select net $i \in \mathcal{N}_{\mathcal{R}}$ if its largest candidate route variable (i.e., highest x_{it} value from the previous LP solution) includes edge e. For the first iteration, use the route provided by the initial solution.
3. Go to the next edge and repeat step 2. Stop as soon as a fixed number of nets (=200) are selected. We determined this value to be suitable in our experiments for all the ISPD'11 benchmarks and our discussed subregion size.

Note, each LP includes all the constraints given in the MILP formulation so all edges in the subregions are included in it. All binary variables are relaxed to be in the interval [0,1]. Overall, the above procedure ensures that a small number of nets are selected in each subregion. The nets are deemed critical because the edges used to identify them have higher congestion as determined by the most-recent LP solution. The LP-solver can in turn more effectively reduce the *PCR* objective in consecutive LPs because congested edges have a higher degree of contribution to the *PCR*.

2) Selecting Candidate Routes: Once the set of critical nets are selected, each LP adds *at most* two candidate routes per critical net.

We first explain how the first candidate route is found for a critical net. This is done by applying decomposition for a multi-terminal net to break it into two-terminal subnets which is done similar to prior works such as [9]. We then use the shortest path (i.e., min-cost path in a weighted graph) to connect each sub-net.

Specifically, we start by assigning a weight w_e for each edge $e \in \mathcal{R}$ in the considered (3D) routing subregion. The weight of each edge is determined based on its utilization which is computed from the most-recent LP solution (or the initial routing solution in the first LP). We then apply a transformation to simplify this 3D graph to a weighted 2D graph in order to reduce its size and accelerate the runtime of the min-cost path problem.

Specifically, for the 3D grid-graph G of size $X \times Y \times L$, the 2D grid-graph G_{2D} will be of size $X \times Y$. To determine the weight of an edge in the G_{2D}, for each edge $e_{2D} = ((i_1, j_1), (i_2, j_2))$ we consider all edges in G specified by $e = ((i_1, j_1, \ell), (i_2, j_2, \ell)), \forall \ell \in L$. The weight of e_{2D} denoted by w_{2D} is given by $w_{e_{2D}} := \min_\ell(w_e)$. A net terminal located at (i,j,ℓ) in G will also map to (i,j) in G_{2D}. For each edge e_{2D} in G_{2D} we also remember the layer ℓ of the corresponding 3D edge in G which had the minimum weight.

Figure 2 shows an example on how this transformation is done together with the layer remembered for each 2D edge.

Once a shortest path is found in G_{2D}, a 3D path is immediately created by assigning each edge to its corresponding layer and then connecting the edges (which have a shared vertex in G_{2D}) with vias.

Theorem 1. *Our transformation guarantees that the shortest-weight path in G_{2D} creates the shortest-weight path on G.*

Proof. Assume there exists a shortest path P in G which includes at least one edge $e = ((i_1, j_1, \ell), (i_2, j_2, \ell))$ in which w_e is not the smallest compared to $e' = ((i_1, j_1, \ell'), (i_2, j_2, \ell'))$ which is above or below it on a different layer. Replace e with e' in P and create a connected path by adding vias from the two end points of e'. The new path P' will have a smaller total weight than P because $0 < w_{e'} < w_e$ and the added vias have a weight of 0. This contradicts the initial assumption that P is the shortest path. □

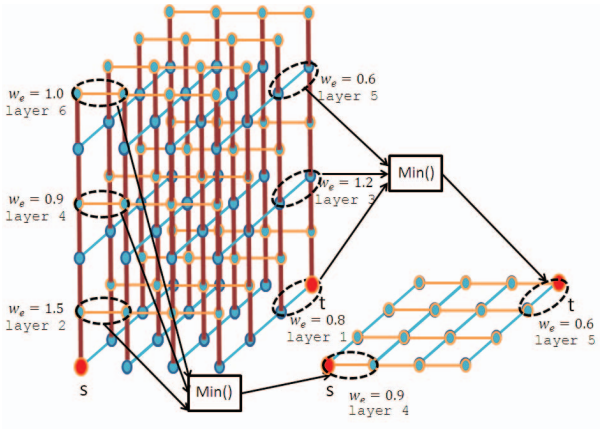

Fig. 2. Creating a 2D grid graph for solving the 3D shortest path problem.

We also consider adding a second candidate route for each selected net as follows. We first identify the "best" candidate route of net i which corresponds to tree t with the highest x_{it} value from the solution of the previous LP iteration. After updating the edge weights at the beginning of the new iteration, we consider moving each flat segment of t to the other layers. If the sum of the edge weights for a segment decreases, then it will be moved. This approach founds a second candidate route for a net if there is at least one such segment that has a decreased cost. Determining if a second candidate route can be found for a net is done extremely fast and in a runtime smaller than solving the shortest path problem.

IV. Simulation Results

A. Simulation Configuration

We implemented our global routing procedure, denoted by IGR, in C++ and ran experiments on a 2.8GHz Intel CPU with 12GB of memory. Experiments were ran on the ISPD'11 routability-driven placement benchmark [11]. We used a placed version for each benchmark by downloading the best placement solution generated for that benchmark from the ISPD'11 contest website.

To create an initial solution for IGR, we used the NCTU-GR 2.0 [6] global router which ran in regular mode. *As a result our initial global routing solutions were already optimized and had a total overflow of 0 in most benchmarks.*

To implement IGR, 3D non-overlapping subregions were formed as described in Section III-A, and covered about 25% of the layout. The global routing problem inside each subregion was defined to optimize the *PCR* objective according to the following penalty function.

$$p_e = \begin{cases} 0 & \text{if } 0 \le r_e < 0.5 \\ 10r_e - 5 & \text{if } 0.5 \le r_e < 0.7 \\ 1000r_e - 698 & \text{if } r_e \ge 0.7 \end{cases}$$

This function assigns a mild penalty to edges with congestion ratio of $0.5 \le r_e < 0.7$. However it assigns a much higher (X10) penalty slope for congestion ratio of 0.7 or above. This makes all the edges with congestion ratio greater than 0.7 to have a higher contribution to the *PCR* objective, thus be more likely to be reduced by IGR. This includes edges with overflow (i.e., congestion ratio above 1) as well as edges with congestion ratio close to 1; as noted in [1], this latter group of edges can easily get over-capacity if rerouting becomes necessary at the detailed-routing stage when additional factors need to be considered which were ignored during global routing. All LPs and the MILPs of subregions were solved using CPLEX V12.6.0 [5].

TABLE I
COMPARISON OF THE DISTRIBUTION OF EDGE CONGESTION RATIO

			Distribution of Edge Congestion Ratio										
Benchmark	**GR**		**0**	**(0.0,0.2]**	**(0.2,0.4]**	**(0.4,0.6]**	**(0.6,0.8]**	**(0.8,1]**	**>1**	**TWL**	**TOF**	**#sub**	**T_{sub}**
superblue1	**INIT**	#edges	529110	485072	380205	328254	312823	327201	0	14735	0		
	IGR	%reduction	1.85%	4.96%	4.84%	-18.21%	-16.53%	18.09%	0.00%	15253	0	227	39
superblue2	**INIT**	#edges	1122420	757910	655549	662891	777935	1216494	239	31385	1732		
	IGR	%reduction	0.61%	3.21%	4.47%	-10.56%	-15.04%	10.40%	17.57%	32181	1732	536	83
superblue4	**INIT**	#edges	129022	198833	196199	199062	214162	366314	52	10781	282		
	IGR	%reduction	2.65%	4.46%	2.22%	-11.48%	-12.76%	9.16%	25.00%	11009	282	121	43
superblue5	**INIT**	#edges	982709	691280	524226	422988	373759	520927	0	17549	0		
	IGR	%reduction	0.77%	4.19%	4.51%	-17.30%	-20.51%	17.20%	0.00%	18235	0	344	64
superblue10	**INIT**	#edges	779358	591978	533919	506525	533167	915525	0	25094	0		
	IGR	%reduction	1.51%	3.66%	1.18%	-13.38%	-12.04%	10.06%	0.00%	25697	0	385	113
superblue12	**INIT**	#edges	250428	148342	125669	148520	227682	794792	0	22450	0		
	IGR	%reduction	1.49%	5.83%	5.93%	-10.76%	-11.62%	2.85%	0.00%	22716	0	143	37
superblue15	**INIT**	#edges	45970	98203	144774	196549	289995	654558	0	17824	0		
	IGR	%reduction	5.45%	7.79%	6.70%	-8.19%	-11.25%	4.41%	0.00%	18087	0	123	41
superblue18	**INIT**	#edges	279609	129569	118442	124688	140766	279642	0	9561	0		
	IGR	%reduction	1.53%	6.47%	8.89%	-15.20%	-17.03%	7.05%	0.00%	9813	0	96	38
Average		%reduction	**1.98%**	**5.07%**	**4.84%**	**-13.13%**	**-14.59%**	**9.90%**	**5.32%**				

B. Comparison of the Distribution of the Congestion Ratios

Table I shows the impact of IGR on the distribution of edge congestion ratios. Columns 4-10 correspond to edges with different ranges of congestion ratio. Different rows show the distribution for the initial solution (expressed in terms of number of edges in each range) as well as of IGR (expressed in terms of % reduction in number of edges in each range compared to the initial). For example for superblue1, there were 327201 edges with congestion ratio in the range of (0.8,1] in the initial solution, and IGR reduced the *number* of these edges by 18.09%.

Within the (0.8,1] range, we were able to further reduce the number of edges within the (0.9,1] sub-range, on-average by 12.83% and at most by 23.80% over all the benchmark.

Columns 11 and 12 report the total wirelength (TWL) and total overflow (TOF). (We verified that our calculation of these quantities matched with the ones generated by the ISPD'11 contest evaluation script for each benchmark.) All TOFs remained unchanged after applying IGR. TWL increased slightly, however TWL is not considered in our formulation and to control long detouring, we ensure the initial routes are decomposed on the subregion boundaries and rerouted only inside each subregion.

The last row shows the average %reduction by IGR for each range over all the benchmarks. On average IGR was able to reduce the number of congested edges which fall in the range (0.8,1] by 9.9%. Only two benchmarks (superblue2 and superblue4) had edges with congestion ratio >1. These are few edges having overflow and IGR associates a higher penalty with them so IGR reduced their count by a higher percentage of 17.57% and 25.00% for superblue2 and superblue4 respectively. In return, IGR increased the edge count falling in the ranges with lower congestion ratio (0.4,0.8].

Column 13 reports the number of subregions created by IGR for each benchmark. Note these are independent and non-overlapping subregions of fixed size 20x20x9 so they can be parallel-processed. Column 14 reports the maximum runtime of a subregion in each benchmark in seconds. The average runtime is less than a minute and the maximum runtime of subregion over all benchmarks was 113 seconds. This runtime includes all steps of the global routing procedure for a subregion.

We note, we consider IGR to be a fast procedure in the presence of parallel cores/machines which is common these days. So the wall runtime of our procedure will also be very fast and comparable to the the maximum runtime of a subregion.

V. CONCLUSIONS AND FUTURE WORK

We introduced IGR, a global routing procedure for controlling the distribution of routing congestion for an input routing solution which has already been optimized for the traditional TOF metric. This was done by solving a mathematical formulation which minimized a user-defined, convex piece-wise linear function that penalized edges within undesirable ranges of congestion. IGR guaranteed that TOF of the input solution will not degrade. Our future plan is to study the impact of various strategies for penalty selection at the global routing distribution, impact of vias and other factors affecting the effort at detailed routing stage.

REFERENCES

[1] C. J. Alpert and G. E. Tellez. The importance of routing congestion analysis. In *DAC.COM Knowledge Center Article*, 2010.

[2] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang. NTHU-Route 2.0: a robust global router for modern designs. *IEEE TCAD*, 29(12):1931–1944, 2010.

[3] M. Cho, K. Lu, K. Yuan, and D.Z. Pan. BoxRouter 2.0: A hybrid and robust global router with layer assignment for routability. *TODAES*, 14(2), 2009.

[4] M.-K. Hsu, S. Chou, T.-H. Lin, and Y.-W. Chang. Routability-driven analytical placement for mixed-size circuit designs. In *ICCAD*, pages 80–84, 2011.

[5] *IBM ILOG CPLEX V12.6.0, User's Manual for CPLEX*, 2013.

[6] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao. Nctu-gr 2.0: Multithreaded collision-aware global routing with bounded-length maze routing. *IEEE TCAD*, 32(5):709–722, 2013.

[7] W.-H. Liu, Y. Wei, C. C. N. Sze, C. J. Alpert, Z. Li, Y.-L. Li, and N. Viswanathan. Routing congestion estimation with real design constraints. In *DAC*, pages 1–8, 2013.

[8] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. *IEEE TCAD*, 27(6):1066–1077, 2008.

[9] H. Shojaei, A. Davoodi, and J. Linderoth. Congestion analysis for global routing via Integer Programming. In *ICCAD*, pages 256–262, 2011.

[10] H. Shojaei, A. Davoodi, and J. T. Linderoth. Planning for local net congestion in global routing. In *ISPD*, pages 85–92, 2013.

[11] N. Viswanathan, C. J. Alpert, C. C. N. Sze, Z. Li, G.-J. Nam, and J. A. Roy. The ISPD-2011 routability-driven placement contest and benchmark suite. In *ISPD*, pages 141–146, 2011.

[12] Y. Wei, C. C. N. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. N. Reddy, A. D. Huber, G. E. Téllez, D. Keller, and S. S. Sapatnekar. GLARE: global and local wiring aware routability evaluation. In *DAC*, pages 768–773, 2012.

[13] T.-H. Wu, A. Davoodi, and J. T. Linderoth. Global routing via integer programming. *IEEE TCAD*, 30(1):72–84, 2011.

[14] Y. Xu and C. Chu. MGR: Multi-level global router. In *ICCAD*, pages 250 –255, 2011.

[15] Y. Xu, Y. Zhang, and C. Chu. FastRoute 4.0: global router with efficient via minimization. In *ASPDAC*, pages 576–581, 2009.