Optimizing Routability in Large-Scale Mixed-Size Placement

Jason Cong[‡†], Guojie Luo[*†], Kalliopi Tsota[‡], and Bingjun Xiao[‡]

[‡]Computer Science Department, University of California, Los Angeles, USA
[*]School of Electrical Engineering and Computer Science, Peking University, Beijing, China
[†]Joint Research Institute in Science and Engineering by Peking University and UCLA
{cong,ktsota,xiao}@cs.ucla.edu; gluo@pku.edu.cn

ABSTRACT

One of the necessary requirements for the placement process is that it should be capable of generating routable solutions. This paper describes a simple but effective method leading to the reduction of the routing congestion and the final routed wirelength for large-scale mixed-size designs. In order to reduce routing congestion and improve routability, we propose blocking narrow regions on the chip. We also propose dummy-cell insertion inside regions characterized by reduced fixed-macro density. Our placer consists of three major components: (i) narrow channel reduction by performing neighbor-based fixed-macro inflation; (ii) dummy-cell insertion inside large regions with reduced fixed-macro density; and (iii) pre-placement inflation by detecting tangled logic structures in the netlist and minimizing the maximum pin density. We evaluated the quality of our placer using the newly released DAC 2012 routability-driven placement contest designs and we compared our results to the top four teams that participated in the placement contest. The experimental results reveal that our placer improves the routability of the DAC 2012 placement contest designs and effectively reduces the routing congestion.

1. INTRODUCTION

One of the most crucial objectives of modern-day VLSI placement is the minimization of the final routed wirelength on the chip. Satisfying this objective has a detrimental effect on the performance of a placer and greatly affects factors such as congestion, delay, and timing. Despite being an important objective, the routed wirelength is usually only taken into account during the post-placement steps by employing techniques for local congestion minimization. However, techniques that are based on local congestion information have a relatively small impact on the quality of the final placement.

A number of approaches have been proposed for reducing routing congestion in placement. The approach in [14] discusses the use of integer linear programming to improve local routing congestion. In [15], the congestion estimation model is based on a stochastic algorithm that computes the horizontal and vertical track usage. The approach in APlace 2.0 [9, 10] describes a congestion estimation method [11] that is based on integrating congestion information into an objective function. The routability-driven placement approach [13] solves a sequence of unconstrained optimization problems, incorporating the wire density as a weighted penalty into an objective function. The authors in [17] examine the correlation between the Steiner tree and the routed wirelength in placement, and propose the ROOSTER algorithm in order to reduce the routed wirelength. The authors of RUDY [20] introduce a model that estimates the wire distribution on the chip by taking into consideration the wire density of each net. The approach proposed in CRISP [18] employs fast global routing to determine congested regions. The approach in [7] optimizes routability using pin density, routing overflow, and macro porosity consideration. The approach proposed in simPLR [12] applies look-ahead routing in a flat placement framework to reduce congestion. The authors of Ripple [5] use cell inflation and net-based movement for routability-driven placement.

The International Symposium on Physical Design (ISPD) 2011 contest [21] was organized to enhance research in VLSI and enable advances in routability-driven placement. The contest also introduced a benchmark suite of industrial ASIC designs that contain placement and routing blockages, varying metal width and spacing, non-rectangular fixed objects, and fixed ′Not in Image′ objects. These objects have pins on metal layers on top of the ones used within cells for internal pins and routing. The Design Automation Conference (DAC) 2012 routability-driven placement contest [22] was a continuation of the ISPD 2011 contest [21] and introduced a benchmark suite of modern industrial ASICs, containing information for both placement and routing. The DAC 2012 contest [22] also introduced a routability metric that describes congestion more accurately than the ISPD 2011 contest metric, as well as two academic global routers for the evaluation of the results. In contrary to the global router [19], that was used to evaluate the routability of the ISPD 2011 contest [21] circuits, the routers NCTUgr and BFG-R of the DAC 2012 contest [22] have improved performance while being machine-independent.

By targeting congestion during global placement we are able to minimize the routing congestion and the final routed wirelength. The remainder of the paper is structured as follows: Section 2 describes our placement flow, while Section 3 introduces the contest metric. Section 4 discusses our proposed technique of neighbor-based fixed-macro inflation to block narrow channels. Section 5 introduces the technique of dummy-cell insertion inside regions of reduced fixed-macro density. Section 6 describes the application of pre-placement inflation in our placer. Section 7 reports our experimental results. Section 8 concludes the paper.

2. PLACEMENT FLOW

We follow an analytical [7, 2, 16] placement approach. To estimate the routing congestion, we first decompose multi-pin nets into two-pin nets by FLUTE [3, 4]. Then, we divide the chip into global tiles and compute the congestion of each tile by taking into account the routing demand and the routing supply (separately for horizontal and vertical directions). As in [5], let $TileWidth$ be the width ($TileHeight$ the height) of the tile, and $WidthBB$ be the width ($HeightBB$ the height) of the bounding box of the two-pin net. Also, let $Ovlp$ be the overlapping area between the tile and the bounding box of the net and $WireH$ be the horizontal ($WireV$ the vertical) wire area of the net. Finally, let $BlockageH$ be the product of the ratio of horizontal ($BlockageV$ the vertical) tracks which are occupied by each routing blockage and the overlapping area between the blockage and the tile. For the horizontal direction:

$$SupplyH = (TileWidth)(TileHeight) - BlockageH,$$

$$DemandH = \frac{(Ovlp)(WireH)}{(WidthBB)(HeightBB)},$$

$$CongestionH = \frac{SupplyH - DemandH}{SupplyH}.$$

For the vertical direction:

$$SupplyV = (TileWidth)(TileHeight) - BlockageV,$$

$$DemandV = \frac{(Ovlp)(WireV)}{(WidthBB)(HeightBB)},$$

$$CongestionV = \frac{SupplyV - DemandV}{SupplyV}.$$

We perform cell inflation to alleviate congested tiles. The inflation pattern is similar to the one described in [5]. In order to further reduce congestion, we propose blocking narrow channels on the chip by inflating fixed-macros that create such channels. We also propose inserting dummy-cells locally on the placement area, inside regions of reduced fixed-macro density. These two methods are applied during the last level of the placement framework. To our knowledge, there are no published papers that apply narrow channel reduction and dummy-cell local insertion to routability-driven placement. Furthermore, we perform pre-placement inflation at each level of the placement framework. In contrast to the in-placement inflation used in [1, 6], our pre-placement inflation takes place before placing the components at each level, and is not iteratively applied during that particular level.

Let $(\mathbf{x},\mathbf{y})$ be the vector of cell coordinates and $HPWL(\mathbf{x},\mathbf{y})$ be the total half-perimeter wirelength on the placement area. We enable cell spreading by applying a uniform placement grid on top of the placement area. We define $D_b(\mathbf{x},\mathbf{y})$ to be the cell density inside placement bin b, and M_b to be the average cell density. The placement formulation is:

$$\min HPWL(\mathbf{x},\mathbf{y})$$
$$\text{s.t.}$$
$$D_b(\mathbf{x},\mathbf{y}) = M_b, \forall \text{ placement bin } b.$$

3. ROUTABILITY METRIC

The metric of DAC 2012 routability-driven placement contest [22] accounts for both routability and runtime. Let ACE be the average congestion of g-cell edges based on the histogram of g-edge congestion as described in [22, 23]. Then, $ACE(x)$ computes the average congestion of the top x% congested g-cell edges. For the contest, the set of ACE values was computed using $x \in 0.5, 1, 2, 5$. In order to simplify calculations, the contest routers NCTUgr and BFG-R report the set of ACE values. Based on the ACE metric, we calculate the peak-weighted congestion as:

$$PWC = \frac{ACE(x)}{4}, x \in 0.5, 1, 2, 5. \quad (1)$$

Using (1), the routing congestion is:

$$RC = \max(100, PWC). \quad (2)$$

Let PF be the penalty factor that scales the $HPWL$ to account for routing congestion. In the DAC 2012 placement contest PF=0.03 so that for every 1% excess routing congestion, there is a 3% wirelength penalty. Also, let $RuntimeFactor$ be the runtime factor that penalizes placers based on their runtime. Using (2), the contest metric is defined as the scaled wirelength of the placement solution as follows:

$$ContestMetric = HPWL(1 + PF(RC - 100))(1 + RuntimeFactor). \quad (3)$$

4. NARROW CHANNEL REDUCTION BY APPLYING NEIGHBOR-BASED FIXED-MACRO INFLATION

A common factor of congestion in modern industrial designs is the existence of fixed macros on the placement area. These macros are responsible for creating narrow channels as illustrated in Figure 1. The existence of narrow channels on the placement area contributes to the routing congestion on the chip. This is due to the fact that cells placed in a narrow regions have difficulty in escaping during placement, thus leading to an increase of the routing congestion in their neighborhood.

We propose blocking narrow channels on the chip in order to reduce routing congestion. Our method is implemented by inflating fixed macros based on their distance to neighboring fixed macros. Our neighbor-based fixed-macro inflation for blocking narrow channels is further described as follows: For each one of the fixed macros, we define a set of right-side neighbors that consists of all fixed macros located on the right side of this particular macro, and whose distance from the macro is smaller than a right-threshold value.

Similarly, we define a set of left-side neighbors of the macro, with distance smaller than a left-threshold value, a set of top-side neighbors of the macro, with distance smaller

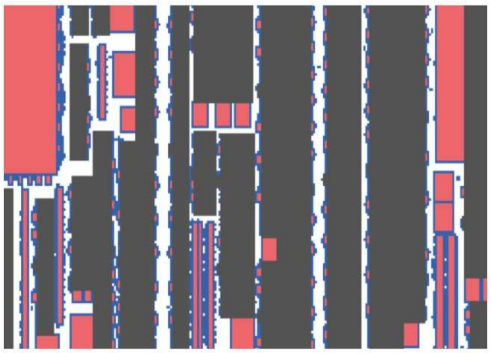

Figure 1: Illustration of narrow channels. Close-up into a region of the floorplan layout of the DAC 2012 routability-driven placement contest design superblue3, obtained from [22]. Light-red shaded boxes correspond to rectangular fixed macros, while gray-shaded boxes correspond to non-rectangular fixed macros.

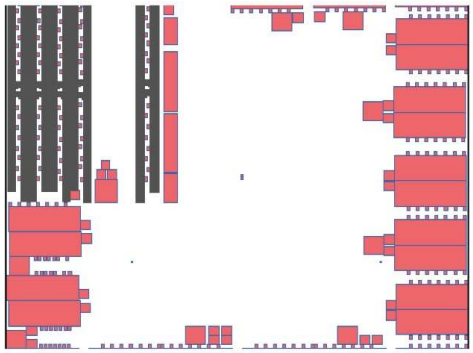

Figure 2: Illustration of second case of congestion. The floorplan layout of DAC 2012 routability-driven placement contest design superblue16 (obtained from [22]) has the majority of its fixed macros located at the periphery. The empty center of the placement area creates routing congestion, since a large number of cells are placed inside this region.

than a top-threshold value, and a set of bottom-side neighbors of the macro, with distance smaller than a bottom-threshold value. The four threshold values, that determine the closest neighbors on each side of the macro, may not necessarily be equal. We also define four rates of inflation for the macro, each rate associated to the degree of inflation we apply to the macro towards the respective side. For example, to completely block a narrow channel on the right side of the macro, we set the right-side rate of inflation of the macro equal to 100%.

5. DUMMY-CELL INSERTION INSIDE REGIONS OF REDUCED FIXED-MACRO DENSITY

The existence of narrow channels on the placement area is not the only factor to take into account in order to improve routability. Another factor is the existence of large empty regions on the design. As illustrated in Figure 2, when a design has the majority of its macros located at the periphery, then the empty part of the placement area may become congested during the placement of cells. To avoid this type of congestion, we identify large empty regions on the chip and insert dummy cells inside them. To identify such regions, we apply a coarse grid on the placement area and determine bins with reduced fixed-macro density. Then, we insert dummy cells inside the bin with the smallest fixed-macro density value. More specifically, after applying a coarse grid on top of the placement area, we calculate the overlap of each fixed macro with the bins of the grid. Then, we identify the bin that has the smallest amount of overlap with fixed macros. Finally, we insert dummy cells in this particular bin.

6. PRE-PLACEMENT INFLATION

To identify regions characterized by routing congestion, we usually employ techniques that are based on pin density, net density, and/or fast routing. Then, we inflate cells located inside congested regions. In order to reduce congestion, we perform pre-placement inflation at each level of the placement framework.

6.1 GTL-Based Inflation

We adopt the Group of Tangled Logic (GTL) metric that was introduced in [8] to detect tangled logic structures in a netlist. Let $T(C)$ be the net cut of cell cluster C, $|C|$ be the number of cells in C, A_C be the average pin count of cells in C, A_G be the average pin count of all cells, and p be the Rent exponent. The GTL score of C is defined as follows:

$$GTL(C) = \frac{T(C)}{A_G |C|^{p\frac{A_C}{A_G}}}.$$

The GTL score curve of C is illustrated in Figure 3 and corresponds to the growth of C. During the growth of a cluster from a seed cell, the score of the cluster is modified by iteratively adding highly connected neighbors. The curve contains a distinct trough if a tangled logic structure appears during the growth. The position of the trough indicates when the cluster growth has reached the most tangled logic structure. The GTL-based technique for cluster growth imposes a significant runtime overhead to the algorithm. In order to find tangled logic structures, the algorithm selects a large number of seed cells and carries out the growth process multiple times. As reported in [8], the runtime for a circuit with 800K vertices may take up to 141 minutes, even if eight parallel threads are used. This particular circuit size only corresponds to the median among the benchmarks of the ISPD 2011 contest [21].

To solve the problem of increasing runtimes we incorporate the cluster growth method to the multilevel placement engine. A bottom-up clustering of cells is initially performed and all cells are treated as objects. The best pair of objects is identified and clustered into a new object and the process continues iteratively until the netlist becomes sufficiently coarse. We integrate GTL scoring into the clustering process with a small runtime overhead and each object is associated to a GTL score curve. When a pair of objects is

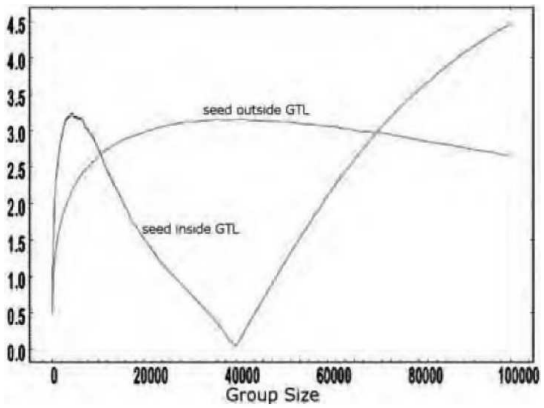

Figure 3: An example of the GTL-score curve [8].

clustered into a new object, the GTL score of the new object is calculated. Compared to the cluster growth method in [8] our method results in a runtime overhead of 40 seconds for a circuit of approximately 800K vertices, a speedup of over 260X.

In [8], all cells inside the detected tangled logic structure are inflated by a factor of four. This type of naive inflation may violate the available whitespace constraint. Also, it may be incapable of reflecting differences among the detected tangled logic structures. Let $TroughWidth(C)$ correspond to the trough width of the GTL score curve of C. In our work, the allocation of the whitespace among the detected tangled logic structures is proportional to their respective weights as follows:

$$Weight(C) = TroughWidth(C)\frac{|C|^2}{Area(C)}.$$

If the GTL score curve of C has a larger trough width, then it has a more distinct trough and C is a more tangled logic structure. The weight of the C is based on the observation that clusters with a large number of cells and smaller area are more likely to be congested and should be inflated.

6.2 Pin Density-Based Inflation

Pin density-based inflation minimizes the maximum pin density. It is based on the fact that inflated cells reserve routing resources proportional to the number of pins they contain. This approach employs a simple model for local nets that consume routing resources.

The algorithm processes the cells in the order of decreasing pin density, then determines how many of these cells can be inflated within the given whitespace budget, so that all inflated cells have the same pin density d_{max}. Let p_i be the number of pins of cell i, $A_i(A_i')$ be the original (inflated) area of cell i, and W be the whitespace allocated for cell inflation. The inflated area $\{A_i'\}$ is the variable in the following optimization problem:

$$\begin{aligned} &\min \max_i \{p_i/A_i'\} \\ &\text{s.t.} \\ &A_i \le A_i', \forall i, \\ &\sum_i (A_i' - A_i) = W. \end{aligned} \tag{4}$$

Although the problem formulation in (4) is nonlinear, it can be proved that Algorithm 1 solves this problem optimally by iteratively assigning whitespace to cells of increased pin density, until there is no whitespace left.

Algorithm 1 Pin Density-Based Inflation.

Input: Pin number $\{p_i\}$ and original area $\{A_i\}$ of cells, available amount of whitespace W for cell inflation.
Output: Inflated areas $\{A_i'\}$ of cells.
Step 1: Sort cells in descending order by making use of their pin density s.t. $p_i/A_i \ge p_{i+1}/A_{i+1}, \forall i$.
Step 2: Find the largest index k s.t. $I_k \le W \le I_{k+1}$, where $I_k = A_k(\sum_{i=1}^{k} p_i)/p_k - (\sum_{i=1}^{k} A_i)$.
Step 3: Compute the maximum pin density after inflation $d_{max} = (\sum_{i=1}^{k} p_i)/(W + \sum_{i=1}^{k} A_i)$.
Step 4: Compute the inflated area $A_i' = \begin{cases} p_i/d_{max} & (i \le k) \\ A_i & (k < i) \end{cases}$.

THEOREM *1. []The solution given by Algorithm 1 is the optimal solution of the nonlinear programming problem* (4).

PROOF. The pin density d_{max} is the maximum pin density after inflation, because

$$W < I_{k+1} \Leftrightarrow W < A_{k+1}\frac{\sum_{i=1}^{k+1} p_i}{p_{k+1}} - \sum_{i=1}^{k} A_i$$

$$\Leftrightarrow W + \sum_{i=1}^{k} A_i < A_{k+1}\frac{\sum_{i=1}^{k} p_i}{p_{k+1}} \Leftrightarrow \frac{W + \sum_{i=1}^{k} A_i}{\sum_{i=1}^{k} p_i} < \frac{A_{k+1}}{p_{k+1}}$$

$$\Leftrightarrow \frac{1}{d_{max}} < \frac{A_{k+1}}{p_{k+1}} \Leftrightarrow \frac{p_{k+1}}{A_{k+1}} < d_{max}.$$

Thus,

$$\begin{cases} p_i/A_i' = d_{max} & (i \le k), \\ p_i/A_i' < p_k/A_k' = d_{max} & (k < i). \end{cases}$$

Assume d_{max} is not the optimal pin density. Then, there is a solution A_i'' s.t. $p_i/A_i'' < d_{max}, \forall i$. However, in such case, the second constraint of (4) is violated, since:

$$\begin{aligned} \left|\sum_{i=1}^{n}(A_i'' - A_i)\right| &= \sum_{i=1}^{k} A_i'' + \sum_{i=k+1}^{n} A_i'' - \sum_{i=1}^{n} A_i \\ &> \sum_{i=1}^{k}\frac{p_i}{d_{max}} + \sum_{i=k+1}^{n} A_i'' - \sum_{i=1}^{n} A_i \\ &= (W + \sum_{i=1}^{k} A_i) + \sum_{i=k+1}^{n} A_i'' - \sum_{i=1}^{n} A_i \\ &= W + \sum_{i=k+1}^{n}(A_i'' - A_i) \ge W. \end{aligned}$$

By contradiction, the solution given by Algorithm 1 is an optimal solution of (4). □

Intuitively, all inflated cells will have the same pin density in an optimal solution; otherwise, if an inflated cell does not achieve maximum pin density, it will be inflated less. The effect of pre-placement inflation is discussed in Section 7. Pre-placement inflation improves the total number of overflows compared to a naive inflation with the same amount of whitespace allocated.

7. EXPERIMENTAL RESULTS

The placers were evaluted on the DAC 2012 routability-driven contest benchmarks, using the contest router NCTUgr [22]. Table 1 reports the scaled wirelength of the contest top four teams Ripple, NTUplace4, mPL12, and simPLR. For each placer, the scaled wirelength is calculated as $HPWL(1+PF(RC-100))$, where PF=0.03. The $HPWL$ is obtained from the output of the global router. Table 2 reports the placement runtimes of the contest top four teams Ripple, NTUplace4, mpl12, and simPLR. For each placer, the runtime is in seconds. For the contest placers, the results were obtained on the contest organizers' 64-bit Intel Xeon CPU X7560 at 2.27GHz (placer binaries were not made available to us). Our placer was evaluated on a machine with the Intel Core i7 Processor at 2.67GHz with 8GB main memory and the average runtime of our placer was estimated by applying a scaling factor.

Our placer minimizes the routing congestion and improves routability. In terms of scaled wirelength, we obtain improved results compared to the winner of the DAC 2012 routability-driven placement contest [22]. In terms of placement runtime, our placer is requires smaller placement runtimes than the analytical placers of the contest. However, no direct comparison can be made due to the fact that the contest binaries were not available to use, therefore we used a different machine for our placer.

8. CONCLUSION

Our proposed placer incorporates narrow channel reduction, dummy-cell insertion inside regions of reduced fixed-macro density, and pre-placement inflation in order to reduce routing congestion and improve the routability of large-scale mixed-size designs. The quality of our placement tool on industrial circuits is evaluated using the global routers of the DAC 2012 placement contest [22] and our results compare favorably to the top four teams that participated in the contest.

9. ACKNOWLEDGMENT

This research was supported in part by the NSF Computing Research Association (Prime Award Number 1136996).

10. REFERENCES

[1] U. Brenner and A. Rohe. An Effective Congestion-driven Placement Framework. In *Proc. ACM/SIGDA International Symposium on Physical Design*, pages 6–11, San Diego, California, Jan. 2002.

[2] T. F. Chan, J. Cong, M. Romesis, J. R. Shinnerl, K. Sze, and M. Xie. mPL6: A Robust Multilevel Mixed-size Placement Engine. In *Proc. ACM/SIGDA International Symposium on Physical Design*, pages 227–229, San Francisco, California, Apr. 2005.

[3] C. Chu. FLUTE: Fast Look-up Table-based Wirelength Estimation Technique. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 696–701, San Jose, California, Nov. 2004.

[4] C. Chu and Y.-C. Wong. FLUTE: Fast Look-up Table-based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design. volume 27, pages 70–83, Jan. 2008.

[5] X. He, T. Huang, L. Xiao, H. Tian, G. Cui, and E. F. Young. Ripple: An Effective Routability-Driven Placer by Iterative Cell Movement. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 74–79, San Jose, California, Nov. 2011.

[6] W. Hou, H. Yu, X. Hong, Y. Cai, W. Wu, J. Gu, and W. H. Kao. A New Congestion-driven Placement Algorithm based on Cell Inflation. In *Proc. Asia and South Pacific Design Automation Conference*, pages 605–608, Yokohama, Japan, Jan. 2001.

[7] M.-K. Hsu, S. Chou, T.-H. Lin, and Y.-W. Chang. Routability-Driven Analytical Placement for Mixed-Size Circuit Designs. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 80–84, San Jose, California, Nov. 2011.

[8] T. Jindal, C. J. Alpert, J. Hu, Z. Li, G.-J. Nam, and C. B. Winn. Detecting Tangled Logic Structures in VLSI Netlists. In *Proc. IEEE/ACM Design Automation Conference*, pages 603–608, Anaheim, California, 2010.

[9] A. B. Kahng and Q. Wang. Implementation and Extensibility of an Analytic Placer. In *Proc. ACM/SIGDA International Symposium on Physical Design*, pages 18–25, Phoenix, Arizona, Apr. 2004.

[10] A. B. Kahng and Q. Wang. Implementation and Extensibility of an Analytic Placer. *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 24(5):734–747, May 2005.

[11] A. B. Kahng and X. Xu. Accurate Pseudo-constructive Wirelength and Congestion Estimation. In *Proc. ACM International Workshop on System-Level Interconnect Prediction*, pages 61–68, 2003.

[12] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov. A SimPLR Method for Routability-Driven Placement. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 67–73, San Jose, California, Nov. 2011.

[13] C. Li and C.-K. Koh. Recursive Function Smoothing of Half-perimeter Wirelength for Analytical Placement. In *Proc. International Symposium on Quality Electronic Design*, pages 829–834, San Jose, California, Mar. 2007.

[14] Z. Li, W. Wu, and X. Hong. Congestion-driven Incremental Placement Algorithm for Standard-cell Layout. In *Proc. IEEE/ACM Asia and South Pacific Design Automation Conference*, pages 723–728, Kitakyushu, Japan, Jan. 2003.

[15] J. Lou, T. S. Krishnamoorthy, and H. S. Sheng. Estimating Routing Congestion using Probabilistic Analysis. In *Proc. ACM/SIGDA International Symposium on Physical Design*, pages 26–33, Phoenix, Arizona, Apr. 2004.

[16] G.-J. Nam and J. Cong. *Modern Circuit Placement:*

Table 1: Experimental Results on the contest placers Ripple, NTUplace4, mPL12, and simPLR. The results for the contest placers were obtained on the contest organizers' 64-bit Intel Xeon CPU X7560 at 2.27GHz. The results of our placer were obtained on an Intel Core i7 Processor at 2.67GHz with 8GB main memory.

	Contest Ripple	Contest NTUplace4	Contest mPL12	Contest simPLR	Our Placer
Circuit	Scaled WL (xE8)	Scaled WL (xE8)	Scaled WL (xE8)	Scaled WL (xE8)	Scaled WL (xE8)
sb19	1.70	1.53	2.46	1.66	1.51
sb14	2.31	2.26	2.67	2.48	2.45
sb16	2.74	2.80	3.01	3.47	2.74
sb9	2.97	2.55	3.22	2.75	2.50
sb3	4.27	3.62	4.66	3.90	3.60
sb11	3.58	3.42	4.52	3.98	3.40
sb6	3.56	3.42	3.95	3.53	3.40
sb2	7.39	6.24	1.33	8.24	6.14
sb12	3.42	3.12	5.40	3.63	3.04
sb7	4.45	3.99	5.24	1.73	3.95
avg.	1.09	1.00	1.41	1.46	0.99

Table 2: Experimental Results on the contest placers Ripple, NTUplace4, mPL12, and simPLR. The results for the contest placers were obtained on the contest organizers' 64-bit Intel Xeon CPU X7560 at 2.27GHz. The results of our placer were obtained on an Intel Core i7 Processor at 2.67GHz with 8GB main memory and the average runtime of our placer was estimated by applying a scaling factor.

	Contest Ripple	Contest NTUplace4	Contest mPL12	Contest simPLR	Our Placer
Circuit	RunTime (s)	RunTime (s)	RunTime (s)	RunTime (s)	RunTime (s)
sb19	2309	8450	11087	981	9911
sb14	2806	9341	10006	1247	7539
sb16	2737	8573	13670	1128	9435
sb9	4307	13129	14910	1821	12736
sb3	5432	14144	19294	2283	12924
sb11	3745	15263	18284	2342	14723
sb6	4944	11179	20508	2484	17121
sb2	6686	17466	23900	3125	18741
sb12	7635	34831	26107	3459	19245
sb7	11285	25983	22233	3025	17243
avg.	2.37	7.24	8.67	1.00	8.07

Best Practices and Results. Springer-Verlag, New York, 2007.

[17] J. A. Roy and I. L. Markov. Seeing the Forest and the Trees: Steiner Wirelength Optimization in Placement. *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 26(4):632–644, Apr. 2007.

[18] J. A. Roy, N. Viswanathan, G.-J. Nam, C. J. Alpert, and I. L. Markov. CRISP: Congestion Reduction by Iterated Spreading during Placement. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 357–362, San Jose, California, Nov. 2009.

[19] H. Shojaei, A. Davoodi, and J. Linderoth. Congestion Analysis for Global Routing via Integer Programming. *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 256–262, Nov. 2011.

[20] P. Spindler and F. M. Johannes. Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement. In *Proc. Design Automation and Test in Europe Conference*, pages 1–6, Nice, France, Apr. 2007.

[21] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy. The ISPD 2011 Routability-Driven Placement Contest and Benchmark Suite. In *Proc. ACM/SIGDA International Symposium on Physical Design*, pages 141–146, Santa Barbara, California, Mar. 2011.

[22] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, and Y. Wei. The DAC 2012 Routability-Driven Placement Contest and Benchmark Suite. In *Proc. IEEE/ACM Design Automation Conference*, pages 774–782, San Francisco, California, June 2012.

[23] Y. Wei, C. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. Reddy, A. D. Huber, G. E. Tellez, D. Keller, and S. S. Sapatnekar. GLARE: Global and Local Wiring Aware Routability Evaluation. In *Proc. IEEE/ACM Design Automation Conference*, pages 768–773, San Francisco, California, 2012.