Modeling and Minimization of Routing Congestion

Maogang Wang

ECE Department
Evanston, IL, 60208, USA

Majid Sarrafzadeh

Northwestern University
e-mail mgwang, majid@ece.nwu.edu

Abstract— Typical placement objectives involve reducing net-cut cost or minimizing wirelength. Congestion minimization is least understood, however, it models routability most accurately. In this paper, we study the congestion minimization problem during placement. First we pointed out that the bounding box router used in [12] is not an accurate measurement of the congestion in the placement. We use a realistic global router to evaluate congestion in the placement stage. This ensures that the final placement is truely congestion minimized. We also proposed two new post processing algorithms, the flow-based cell-centric algorithm and the net-centric algorithm. While the flow-based cell-centric algorithm can move multiple cells at the same time to minimize the congestion, it suffers large consumption of memory. Experimental results show that the net-centric algorithm can effectively identify the congested spots in the placement and reduce the congestion. It can produce on an average 7.7% less congestion than the method proposed in [12]. Finally, we use a final global router to verify that the placement obtained from our algorithm has 39% less congestion than a wirelength-optimized placement obtained by TimberWolf (commercial version 1.3.1).

I. Introduction

Automated cell placement for VLSI circuits has always been a key factor for achieving designs with optimized area usage, wiring congestion and timing behavior. As technology advances, the congestion problem becomes more and more important. With the advent of over-the-cell routing, the goal of every place and route methodology has been to utilize area to prevent spilling of routes into channels. It is this overflow of routes that accounts for an increase in area. The multiple routing layers have enough routing resources to route most wires as long as there are not too many wires congested in the same region. Excessive congestion will result in a local shortage of the routing resource.

Typical placement objectives involve reducing net-cut costs or minimizing wirelength. Because of its constructive nature, min-cut based strategies minimize the number of net crossings but fail to uniformly distribute them [8]. Congestion-driven placement based on multi-partitioning was proposed in [5]. It uses the actual congestion cost calculated from pre-computed Steiner trees to minimize the congestion of the chip, however, the number of partitions is limited due to the excessive computational load. The use of minimal wirelength as a metric to guide placement has been successful in achieving good placement. However, it only indirectly models congestion and the behavior of the router. Reducing the global wirelength helps reduce the wiring demand globally, but does not prevent existing local congested spots. It is entirely feasible for a minimum wirelength solution to require more routing resources through a region than are available. Therefore, traditional placement schemes which are based solely on wirelength minimization [9, 10, 3, 1, 11] cannot adequately account for congestion [8, 5, 7].

In [12], the authors used the definition of overflow to quantitatively represent the congestion. A placement with a smaller overflow value is considered less congested. The authors also used a simple bounding box model to estimate all the routes of nets in placement. Since [12] did not provide any real routing results, it is not clear that the congestion-optimized placement using this simple model [12] is indeed easier to route for a real router.

In this paper, we implement a global router to test the routability of a placement. With this router, we can provide accurate information to know which placement is indeed less congested. We show that the simple bounding box routing estimation [12] is not accurate enough to yield truely congestion-optimized placement results. We use a realistic global router to evaluate congestion in the placement stage. This ensures that the final placement is truely congestion minimized. We also proposed two new post processing algorithms, the flow-based cell-centric algorithm and the net-centric algorithm. While the flow-based cell-centric algorithm can move multiple cells at the same time to minimize the congestion, it suffers large consumption of memory. Experimental results show that the net-centric algorithm can effectively identify the congested spots in the placement and reduce the congestion. It can produce on an average 7.7% less congestion than the method proposed in [12]. At the end, we use a global router to verify that the placement coming out of our congestion minimized stage indeed causes less congestion than a traditional wirelength minimized placement (e.g.,

using TimberWolf).

The rest of the paper is organized as follows: In Section 2, we define the congestion problem and the overflow congestion objective. In Section 3, we introduce our global router and experimentally verify that the simple bounding box routing model does not correlate with the real routing model very well. In Section 4, we analyze the wirelength behavior in a congestion minimization stage. In Section 5, we propose two effective post processing algorithms to minimize the congestion in the layout. The experimental results are shown in Section 6 and the conclusion is in Section 7.

II. Definition of Congestion

A poorly placed layout could be un-routable due to the limited routing resources. In order to ensure the maximum routability in the layout, we should consider congestion minimization in the placement stage.

In this paper we assume that we are given a netlist that consists of a set of cells connected by a collection of nets. Each net consists of a set of pins. Pins are to be assigned a geometric location on the layout surface in the placement process.

In [12], a global bin based definition of congestion was used. This definition is consistent with global routability of layout. Since it is easy and clear to understand, we will use this definition in this paper.

The congestion cost is defined based on the global bin concept. We partition a given chip into several rectilinear regions, each of these regions is called a global bin or a bin for simplicity. Figure 1 shows an example. In Figure 1, we have $4 \times 4 = 16$ global bins. The congestion is quantified as number of crossings between routed nets and global bin edges.

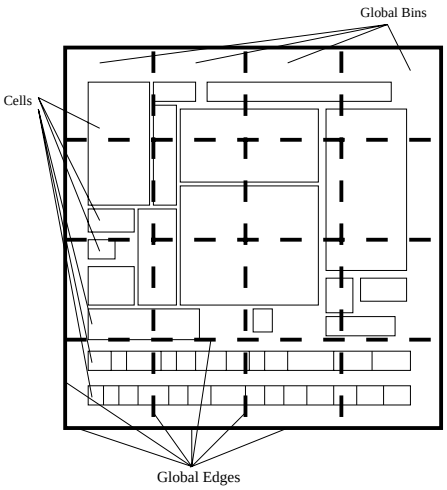

Fig. 1. Layout of a circuit and global bins.

Given a placement, all the cells and pins have fixed positions on the chip. Based on these pin locations, we can globally estimate the route for each net. Therefore, for each global edge e, the routing demand of e, d_e, is defined as the number of the nets crossing e. The routing supply of a global edge e, s_e, is a fixed value which is a function of the length of the edge and technology parameters. A global edge e is congested if and only if the routing demand (number of the crossing nets) exceeds the routing supply of that edge ($d_e > s_e$). If a global edge e is congested, the overflow of e is defined as the exceeding amount of the routing demand over the routing supply of e. The overflow of e is zero if e is not congested.

We will use the congestion overflow as the objective to judge the amount of congestion in a layout. The congestion overflow of a layout is defined as the summation of the overflow for all global edges. The amount of the congestion overflow is the amount of total shortage of routing resources in the layout. Thus a placement with less overflow is less congested.

In [12], the authors used a simple bounding box routing model to calculate the overflow value in placement. Thus it is not clear that the overflow value calculated in this way correlates with the final routability. In next section, we will use a global router to experimentally address this concern.

III. Different Incremental Global Routing Estimation Models

When we are doing congestion minimization, we need to estimate congestion of placement incrementally. Based on the definition of the congestion, a routing model is needed to route all the nets. In this subsection, we will discuss two incremental routing estimation models, one simple model and a more accurate one (both model have been studied extensively in the past).

The first routing model can be best described as a "bounding-box model". It is very simple and fast. Figure 2 shows a sample to-be-routed net which contains five terminals (represented by black solid dots in Figure 2). This method is shown in Figure 2a. Given locations of all the terminals of a net, first we find the bounding box of the net. Then the actual route will be either the upper L-shape half or the lower L-shape half of the boundary of the bounding box determined in a probabilistic manner. This method will ignore terminals in the middle of the bounding box for nets which have more than two terminals.

The second model is a real global routing model. "Real" means that the model uses the same algorithm used at the routing stage after placement. Routing is a relatively well understood problem. The Steiner tree model and the maze routing technique are usually used in the routing stage. Thus we will use the same model (Steiner tree and maze routing) for the incremental congestion estimation. This model will provide a very accurate congestion estimation during the placement stage. However, it is more

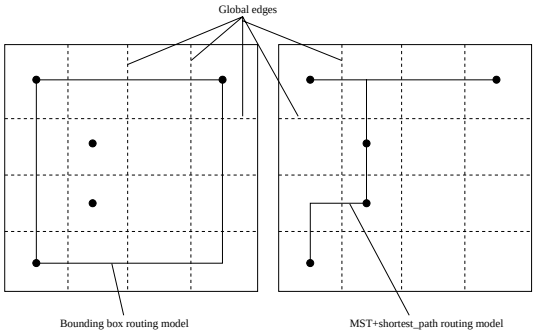

Fig. 2. Two global routing models.

time consuming than the bounding box routing model.

We first want to know that if the bounding box routing model is good enough to be used in the placement stage. For instance, although the absolute congestion value estimated by the bounding box model is in-accurate, if it is correlated to the real congestion value, then we can still use it in the optimization.

We conducted an experiment to test the correlations between the incremental congestion estimation value and the real congestion value provided by a final global router after placement. Our final global router is implemented using the minimum spanning tree (MST) and the shortest path algorithm. This is a slightly simpler version of the Steiner tree and the maze routing algorithm which is widely used in industry. In the routing stage, the procedure of "rip-up and re-route" plays a very important role to reduce congestion. Basically, after we finished routing all the nets in the circuit, we use this procedure to re-route those nets which went through congested regions in previous routes. We have implemented this rip-up and re-route feature in our global router.

The detailed implementation of our router is as follows (and is very similar to router introduced in [6]: Given a placement, we first partition the chip into a set of global bins. Then for each multi-terminal net, we use the MST algorithm to find out which pairs of terminals are to be connected. Then we use the shortest path algorithm to find the shortest route between them. After all the nets are routed, we perform the rip-up and re-route procedure to further improve the global routing result. We sequentially remove each net and re-route it to reduce the overall congestion on the chip. This rip-up and re-route procedure will be repeated until we cannot reduce congestion anymore.

Our incremental accurate estimation model also uses the MST and the shortest path algorithm to be consistent with the final global router. In the incremental estimation context, usually we only make local adjustments to the placement, thus only a small number of nets need to be re-routed to make the incremental estimation. Therefore, we do not need to perform rip-up and re-route on all the nets each time we make a local adjustment However, after a number of local adjustments being made to the placement, we need to perform rip-up and re-route to update the optimal routing order of all the nets.

In experiments, we start from a wirelength optimized placement, each time we (to mimic a simulated annealing algorithm) select a cell and move it to another location. We will estimate the change in the congestion overflow caused by each move using two incremental routing models independently. Finally we compare the two estimation values with the real change on the congestion overflow provided by the final global router. Figure 3 shows the experimental result for the first one hundred moves. The solid line shows the real congestion overflow value obtained from the final global router. The triangle line shows the estimated overflow from the accurate routing model without performing rip-up and re-route. The + line shows the estimated overflow from the bounding box routing model. We can see that the accurate model can almost estimate the overflow value exactly within about the first twenty moves. Thus in the real application, we choose to perform the rip-up and re-route every twenty moves.

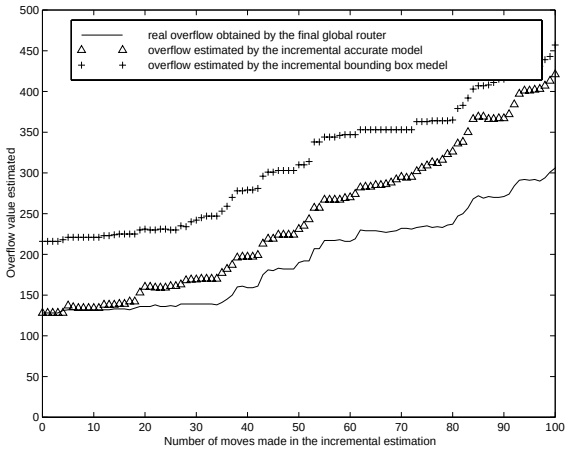

Fig. 3. Correlation results of two estimation models.

Another correlation test is to compare the overflow value estimated by the bounding box router and the accurate router. We generate six different placements (A, B, C, D, E and F). Table I shows the results. This experiment clearly shows that the bound box router **does not correlate** with the real router. For instance, the bounding box router shows that placement A is better than placement B ($14 < 36$). However, the real router shows the opposite ($27 > 9$). Similar examples can also be found in other testing results. Therefore, we cannot use the simple bounding-box routing model in the placement optimization. We should use the same routing model in the placement optimization as the model we used in the final routing stage.

RoutingModel	A	B	C	D	E	F
BBox	14	36	26	27	40	30
Real	27	9	7	4	5	4

TABLE I
CORRELATION TEST BETWEEN THE BOUNDING-BOX AND THE REAL ROUTING MODEL FOR PRIMARY1

IV. Behavior of Wirelength in a Congestion Minimizatio Stage

In [12], the authors revealed that minimizing wirelength is equivalent to minimizing the averge congestion. However, wirelength and congestion can be inconsistent in local regions.

Figure 4 shows an example that minimizing congestion is not equivalent to minimizing wirelength. The sameple circuit contains eight cells and four nets. Among these eight cells, four have no nets attached to them and the other four are circularly connected by four nets as shown in Figure 4. Assume that the wiring supply on each global edge is one, the left part of Figure 4 shows the congestion optimal placement. In this placement, four nets are evenly distributed on the chip which result in a zero overflow (routable) solution. In wirelength optimization, we tend to put as many nets as possible into the same region. The right part of Figure 4 shows the wirelength optimal placement. Since each global bin can only contain two cells, we put four cells along with four nets into two global bins. This results in a wiring demand of two on one global edge. Since the wiring supply is only one, we have overflow of one in this placement.

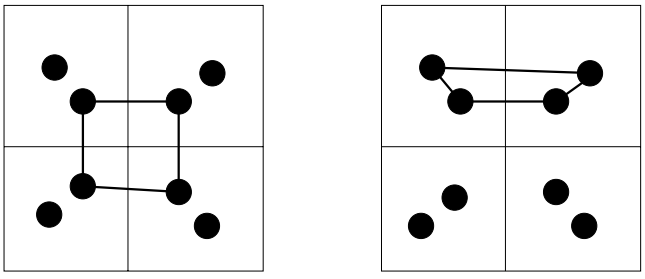

Fig. 4. Minimizing congestion is not equivalent to minimizing wirelength.

The same story can happen in a real circuit (as has been observed by anyone who has worked on the placement problem or has used a placer). As shown in Figure 5, when minimizing wirelength, we tend to put cells within a highly connected cluster close to each other. On the other hand, when minimizing congestion, we tend to banlance all the wires to avoid local congested spots. Thus we might spread out the highly connected clusters slightly to reduce congestion. Therefore, minimizing wirelength and minimizing congestion may conflict each other in local regions. In order to get a congestion optimal placement, we might have to sacrifice wirelength.

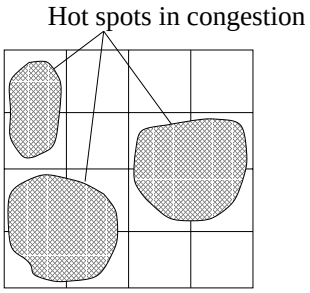

Fig. 5. Minimizing wirelength might create hot spots in congestion.

V. Post Processing Congestion Reduction Algorithms

In [12], the authors used a greedy algorithm to reduce the congestion in a placement. The algorithm try to randomly move a cell to another location. The cell is actually moved if this move results in a reduction in the congestion overflow. We call this blind cell-based strategy the "greedy cell-centric" algorithm. The greedy feature of this algorithm makes it easy to get stuck into a local minimum. To solve this problem, we propose a multiple cell moving strategy based on a net-work flow method. This algorithm is very greedy. It does not have the ability to know where the congestion is and how to reduce it. To improve, we propose a net moving strategy which can identify the highly congested spots and try to move nets out of these spots. The following are two new post processing algorithms we propose:

1. Flow-based cell-centric algorithm: This algorithm uses a flow-based approach to move multiple cells simultaneously.

2. Net-centric algorithm: This algorithm first sort all the nets based on their contribution of congestion. Then it tries to move the net one by one to reduce congestion.

In the flow-based cell-centric algorithm, we try to find better locations for cells to reduce congestion. This can be viewed as a transportation problem. In the corresponding transportation problem, the source of the transportation is all the cells and the destination is all the global bins. A transportation cost is associated with a cell move. We then simultaneously transport the cells to new locations that minimizes the transportation cost. The transportation problem can be transformed into a minimal-cost maximum flow problem [4] on a network. By using the flow

augmentation method [2, 4], we can get a new location assignment of cells with minimum total transportation cost.

Given a placement, we first route all nets. Then we assign a weight to each net. The weight of a net is equal to the number of overflowed global edges the net crosses. We sort the nets in descending order according to their weights. The net with the greatest weight is the one which contribute the most to the total overflow. Thus move this net will most likely to help reducing the congestion. In order to move a net, we consider moving all cells connected to the net. The destination of the move could be any global bin. Thus we look at all the cells connected to the net and move a cell to a new position which can result in a reduction in the congestion overflow. After all the nets have been tried, we will update the net weights according to the new global routing information. We will repeat the above procedure until the congestion overflow cannot be further reduced. Since congestion is essentially produced by nets, moving nets out of the congested region makes more sense than greedily moving single cells.

VI. Experiments

We use the two-stage congestion reduction flow as proposed in [12]. We run simulated annealing using the wirelength objective in the first stage. The output placement from the first stage will be the input to the post processing stage.

We use eight MCNC standard-cell benchmark circuits. The characteristics of these circuits are shown in [12]. The size of the global bin grid is chosen so that each bin has roughly $10 - 50$ cells.

Table II shows the results from the post processing stage with the high quality placements. The *before PP* column in the table is the results before the post processing stage. The percentage improvement column is the improvement of using the net-centric algorithm comparing to the results using the greedy cell-centri algorithm [12]. While all three post process methods can effectively reduce the congestion. We get an average 7.7% improvement comparing to the post processing method proposed in [12]. The flow-based cell-centric algorithm can produce good results. However, due to the large consumption of memory, it failed to run on large circuits.

The whole purpose of reducing congestion in placement is to make the placement easier to route. We use the global router which is introduced in Section 3 to verify this fact. We do global routing on a wirelength optimized placement generated by TimberWolf (commercial version 1.3.1) and compare the congestion of TimberWolf's placement to a congestion optimized placement generated by our approach. Table III shows this comparison. We compare the final overflow value of a wirelength optimized placement generated by TimberWolf and a congestion optimized placement generated by our post processing stage. The average improvement on congestion is about 40%.

Ckts	**before PP**	**cell -cen.**	**flow-based**	**net-cen.**	**net-cen.vs. cell-cen.**
hway2	7	7	7	7	0%
fract	9	9	9	9	0%
p1	111	105	103	90	14.3%
p2	129	103	99	89	13.6%
struct	79	69	58	57	17.4%
biomed	442	360	353	347	3.6%
avqs	224	149	*	131	12.1%
avql	430	323	*	298	7.7 %
ave.					8.6%

TABLE II
Post processing results using different algorithms.

(* out of memory)

VII. Conclusion

In this paper, we pointed out that the bounding box router used in [12] is not an accurate measurement of the congestion in the placement. We use a realistic global router to evaluate congestion in the placement stage. This ensures that the final placement is truely congestion minimized. We also proposed two new post processing algorithms, the flow-based cell-centric algorithm and the net-centric algorithm. While the flow-based cell-centric algorithm can move multiple cells at the same time to minimize the congestion, it suffers large consumption of memory. Experimental results show that the net-centric algorithm can effectively identify the congested spots in the placement and reduce the congestion. It can produce on an average 8.6% less congestion than the method proposed in [12]. Finally, we use a final global router to verify that the placement obtained from our algorithm has 39% less congestion than a wirelength-optimized placement obtained by TimberWolf (commercial version 1.3.1).

TestCase	**ovrflw TW**	**ovrflw opt plcmt**	**% imp. vs. TW**
hway2	12	7	41.7%
fract	35	9	74.2%
p1	159	90	43.4%
p2	128	89	30.4%
struct	79	57	27.8%
biomed	442	347	21.5%
avqs	224	131	41.5%
avql	430	298	30.7%
ave.			38.9%

TABLE III
The final overflow value comparison between different placements.

References

[1] H. Eisenmann and F. M. Johannes. "Generic Global Placement and Floorplanning". In *Design Automation Conference*, pages 269–274. IEEE/ACM, 1998.

[2] L.R. Ford and D.R. Fulkerson. *Flows in Network*. Princeton, NJ, 1962.

[3] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization". *IEEE Transactions on Computer Aided Design*, 10(3):365–365, 1991.

[4] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, 1990.

[5] S. Mayrhofer and U. Lauther. "Congestion-Driven Placement Using a New Multi-Partitioning Heuristic". In *International Conference on Computer-Aided Design*, pages 332–335. IEEE/ACM, November 1990.

[6] R. Nair. "A Simple Yet Effective Technique for Global Wiring". *IEEE Transactions on Computer Aided Design*, CAD-6(2):165–172, March 1987.

[7] P. N. Parakh, R. B. Brown, and K. A. Sakallah. "Congestion Driven Quadratic Placement". In *Design Automation Conference*, pages 275–278. IEEE/ACM, 1998.

[8] Y. Saab. "A Fast Clustering-based Min-cut Placement Algorithm with Simulated-annealing Performance". *VLSI Design: An International Journal of Custom-Chip Design, Simulation, and Testing*, 5(1):37–48, 1996.

[9] M. Sarrafzadeh and M. Wang. "NRG: Global and Detailed Placement". In *International Conference on Computer-Aided Design*. IEEE, November 1997.

[10] C. Sechen. *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer, B. V., Deventer, The Netherlands, 1988.

[11] P. R. Suaris and G. Kedem. "Quadrisection: A New Approach to Standard Cell Layout". In *Design Automation Conference*, pages 474–477. IEEE/ACM, 1987.

[12] M. Wang and M. Sarrafzadeh. "Behavior of Congestion Minimization During Placement". In *International Symposium on Physical Design*, pages 145–150. ACM, April 1999.