Congestion Driven Incremental Placement Algorithm for Standard Cell Layout

Zhuoyuan Li, Weimin Wu, Xianlong Hong
Department of Computer Science and Technology
Tsinghua University, Beijing, 100084, P.R.China

Abstract Congestion minimization is the least understood in placement objectives, however, it models routability most accurately. In this paper, a new incremental placement algorithm C-ECOP for standard cell layout is presented to reduce routing congestion. Congestion estimation is based on a new routing model and a more accurate cost function. An integer linear programming (ILP) problem is formulated to determine cell flow direction and avoid the conflictions between adjacent congestion areas. Experimental results show that the algorithm can considerably reduce routing congestion and preserve the performance of the initial placement with high speed.
Key words: congestion, standard cell, incremental placement

1. Introduction

As VLSI technology advances, the system complexity continues to increase and physical design is getting more and more difficult. With the advent of overcell routing, the goal of every place and route methodology has been to utilize all available active area to prevent spilling of routes into channels. It is the overflow of routes that account for an increase in area. Further heuristic method should be applied in placement to manage local congestion to enhance and improve the latter route ability.

Traditional placement objectives involve reducing net-cut costs or minimizing wire length [1-2]. Because of its constructive nature, min-cut based strategies minimize the number of net crossings but fail to distribute them uniformly [3]. For the same reason, traditional placement schemes which are based mainly on wirelength minimization can not adequately account for congestion. Reducing net-cut and minimizing wirelength only help reduce the routing demand globally but do not prevent causing local routing congestion. How to estimate and reduce congestion in placement is not well studied. Congestion-driven placement based on multi-partitioning was proposed in [4]. It uses the actual congestion cost calculated from precomputed Steiner trees to minimize the congestion of the chip. However, the number of partitions is limited due to the excessive computational load.

Wang et al. [8-10] proposed a consistent routing model defined by demand/supply relationship. Experimental results show that the congestion objective is very ill behaved. So it adapts a post processing approach after placement to reduce congestion. But the demand/supply congestion model and bounding-box routing estimation is too simple and will affect the final result. Since congestion and wirelength are globally consistent, Jason et al. [11-12] considered improving local congestion with incremental placement. However, how to minimize the wirelength changes caused by congestion reducing and maintain the metrics of the initial placement is very difficult.

In this paper, an incremental placement algorithm C-ECOP for improving local congestion is proposed. It first estimates the routing congestion through a new route model. Then it constructs an integer linear programming (ILP) to move cells to reduce congestion. Finally it adjusts the positions of cells to resolve overlap. The rest of this paper is organized as follows. Section 2 describes the routing estimation and congestion measurement used in this work. The algorithm C-ECOP is presented in section 3. Section 4 gives the experimental results to show the effectives of our algorithm. Section 5 is the conclusion.

2. Congestion Estimation in Placement

2.1 Congestion Cost

The congestion cost is defined based on the global bin concept. We partition a given chip into several rectangular regions, each of which is called a global bin. The boundaries of global bins are called global bin edges as shown in Fig.1. The congestion is "related" to the number of crossings between routed nets and global bin edges.

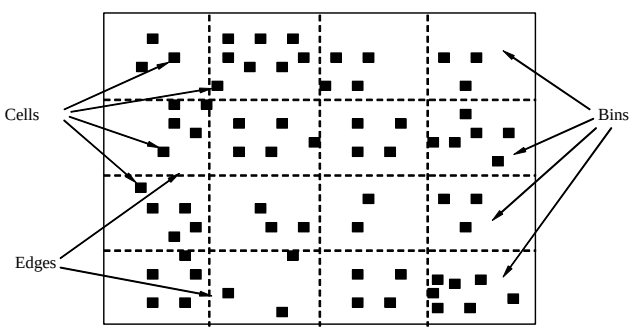

Fig.1 Bin Structure

Given a detailed placement, all the cells and pads have fixed positions on the chip. We can use a "router" to route all the nets. The router can be a very simple global router or even a bounding box router. Therefore, for each global edge e, the routing demand of e, d_e, can be defined as the number of the nets crossing e. The routing supply of e, s_e, is known easily from the technology parameter. A

* The work was supported by the National Natural Science Foundation of China (NSFC) 60121120706, the National Natural Science Foundation of USA (NSF) CCR-0096383, Hi-Tech Research & Development (863) Program of China 2002AA1Z1460 and The National Foundation Research (973) Program of China G1998030403

global edge is congested if and only if routing demand d_e exceeds the routing supply s_e. The overflow is defined as follows:

$$overflow\ (e) = \begin{cases} 0 & (d_e \le s_e) \\ d_e - s_e & (d_e > s_e) \end{cases} \quad (1)$$

Then estimating the congestion of a global bin can be replaced by computing the total overflow of the global edges as (2) and cost function is defined as (3). Most of the algorithms for reducing congestion [6-10] estimate congestion like this.

$$con\,(b_{ij}) = \sum_{k=1}^{4} overflow\ (e_k) \quad (2)$$

$$\cos t_c = \sum_{i=1}^{n}\sum_{j=1}^{n} con\,(b_{ij}) \quad (3)$$

This method to estimate congestion is ill behaved. It can only detect the congestion area that is crossed by a global edge. Actually, the congestion area can be in any place on the chip and it may just locate inside a global bin. Table 1 gives the result of the congestion estimation on a circuit CNT100 with the method above. The chip is partitioned into 4x4, 5x5, 6x6 and 7x7 global bins. It is shown that the total number of congest areas (TNC.), the total demand (TD.) changed due to different partition. The max demand (MD.) even increases when the partition amount increases from 4x4 to 5x5. Because the global edges become shorter, it seems that the routing demand crossing the edge should decrease.

Table 1. Estimation of Congestion on CNT100

Grids	Supply (H/V)*	TNC. (H/V)	TD. (H/V)	MD. (H/V)	MO.** (H/V)
4 x 4	19/12	0/1	261/198	**18/13**	0/1
5 x 5	15/10	4/4	269/192	**20/14**	5/4
6 x 6	13/8	2/8	322/226	**17/11**	4/3
7 x 7	11/7	4/6	391/251	**18/8**	7/1

*H: Horizontal V: Vertical **MO.: Max Overflow

From the table we can see that the total demand increases when partition amount increases. It is because that the nets inside a larger global bin may become cross a global edge when the chip be partitioned with smaller bins. So the vertical and horizontal congestion estimation of a global bin in our algorithm is defined as follows:

$$con_{\,v}(b_{ij}) = \omega_1 r_v(b_{ij}) + \omega_2 \sum_{k=1}^{2} overlfow_{\,v}(e_k)$$

$$con_{\,h}(b_{ij}) = \omega_1 r_h(b_{ij}) + \omega_2 \sum_{k=1}^{2} overlfow_{\,h}(e_k) \quad (4)$$

where r_v and r_h are the vertical and horizontal routing demand inside b_{ij}. It can be obtained by the routing model described in 2.2. ω_1 and ω_2 are the weights.

2.2 Routing Estimation Model

When we are performing congestion reducing, we need to estimate congestion of placement incrementally. A global router is needed here. Obviously, the more accurate is the router, the more accurate is the estimation at the placement stage. Routing with a real global router will provide an accurate congestion estimation. But it will be very time consuming and could not be applied in incremental placement algorithm. Routing with a simple routing model such as the bounding box model will be very fast. But the bounding box model may be far different with the characteristics of the detailed router so that it causes a bad estimation. Wang et al. verify the bounding box model in [8] and prove that it does not correlate with the real router and could not be applied. So it is critical that the algorithm for this application is accurate while maintaining computational efficiency.

A new star model proposed in [6] is used here. It first computes and adjusts the coordinate of the net center. The vertical and horizontal possible route paths connecting each cell to the center are on the edges of a rectangle whose two vertexes locate on the cell and the center. Route possibility on each path is 0.5. Then all the route possibility on the same path is added up and could not exceed 1 as shown in Fig. 2. Experimental results show that the new star model is very close to the real routing in practice [6].

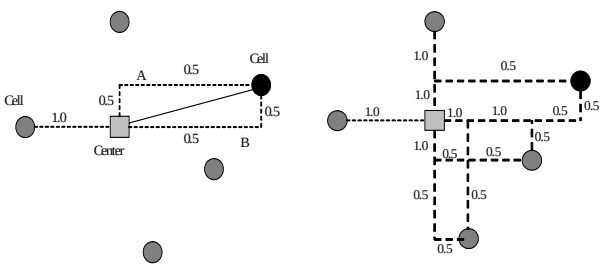

Fig.2 Routing Estimation with new star model

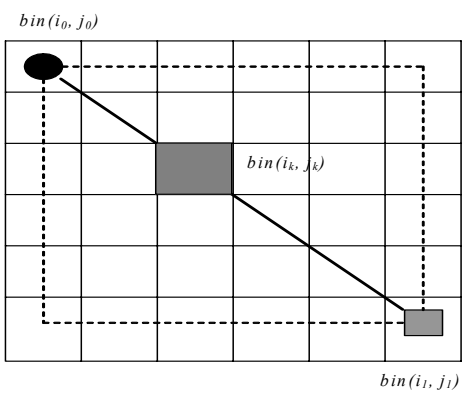

Fig.3 Route Probability

This model has an obvious drawback. The possible route path between a cell and the net center does not necessarily locate on the rectangle edges. It may cross any

bin edges inside the rectangle as shown in Fig.3.

If cell C_k is in $bin(i_0, j_0)$, the net center is in $bin(i_1, j_1)$, all the possible routes crossing the left edge of $bin(i_k, j_k)$ are those from $bin(i_0, j_0)$ to $bin(i_k\text{-}1, j_k)$, crossing the edge between $bin(i_k\text{-}1, j_k)$ and $bin(i_k, j_k)$, then from $bin(i_k, j_k)$ to $bin(i_1, j_1)$. It is in the symmetric form on the right edge of $bin(i_k, j_k)$. So the route possibility crossing the left and right edges of $bin(i_k, j_k)$ could be denoted as:

$$p_l(k) = \begin{cases} w_{kl} & i_0 \le i_k \le i_1 \\ & j_0 \le j_k \le j_1 \\ 0 & otherwise \end{cases} \quad (5)$$

$$p_r(k) = \begin{cases} w_{kr} & i_0 \le i_k \le i_1 \\ & j_0 \le j_k \le j_1 \\ 0 & otherwise \end{cases}$$

$$w_{kl} = C_{i_k - i_0 - 1 + j_k - j_0}^{i_k - i_0 - 1} \times C_{i_1 - i_k + j_1 - j_k}^{i_1 - i_k} / C_{i_1 - i_0 + j_1 - j_0}^{i_1 - i_0} \quad (6)$$

$$w_{kr} = C_{i_k - i_0 + j_k - j_0}^{i_k - i_0} \times C_{i_1 - i_k - 1 + j_1 - j_k}^{i_1 - i_k - 1} / C_{i_1 - i_0 + j_1 - j_0}^{i_1 - i_0} \quad (7)$$

And the possibility of crossing the top and bottom edges, which are denoted as $p_t(k)$ and $p_b(k)$, are computed in the symmetric form. The vertical and horizontal routing demand inside a global bin can be easily known from this approach. The running time of congestion estimation on some circuits through this approach is listed in Table 2. It is shown that this approach is so fast that it could be used in our algorithm.

Table 2. Running Time on Routing Estimation

Circuits	#cells	#nets	Grids	Running time(s)
Ibm01	12,036	11,507	30 x 30	0.57
Ibm02	19,062	18,429	40 x 40	1.95
Ibm03	21,924	21,621	50 x 50	2.05
Ibm04	26,346	26,163	50 x 50	1.93
Ibm05	28,146	28,446	60 x 60	4.48

3. Congestion Reducing through ILP

3.1 Overview

Generally, minimizing congestion and minimizing wirelength conflict each other in local regions. The reducing of congestion means to sacrifice the wirelength. Incremental placement algorithm should achieve trade-off between congestion reducing and preserving the metrics, e.g. wirelength, of the initial placement.

The design flow is shown in Fig.4. The flow tendency of each cell is computed based on force driven by nets. Then cells could move due to the tendency to reduce congestion. An integer linear programming is formulated to deal with the conflicts between multiple congested regions. After that a post process is carried out to place the moved cells and resolve overlap. The iteration of the ECO flow stops when the congestion result is acceptable.

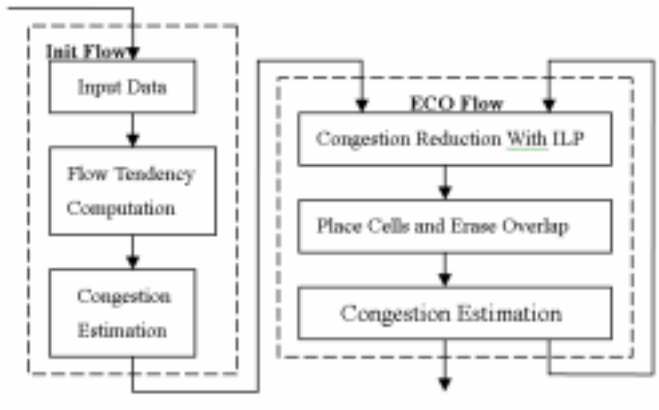

Fig.4 C-ECOP Algorithm

3.2 Cell Flow Tendency Computation

Based on routing estimation, we can identify the congestion bins on the chip. Each global bin whose congestion cost defined by (4) is greater than a certain threshold value or at least one of its global edge is overflowed is considered to be a congest bin. Cells in congest bins should move outside to decrease routing demand and achieve more routing resource. Which cells can be moved out so that the perturbation to the initial placement could be minimized is the key problem. We compute the flow tendency of each cell to deal with this.

The horizontal flow tendency of a cell C_k in *bin(i, j)* is computed as follows:

$$xflow_l(c_k) = \sum_{net_{k1}} p_l(k) - \sum_{net_{k2}} p_r(k) - \sum_{net_{k3}} 1 \quad (8)$$

$$xflow_r(c_k) = \sum_{net_{k2}} p_r(k) - \sum_{net_{k1}} p_l(k) - \sum_{net_{k3}} 1 \quad (9)$$

where net_{k1} are the nets whose center are in bins on the left, net_{k2} are the nets whose center are in bins on the right, net_{k3} are the nets whose center are in bin *b(i, j)*. $p_l(k)$ and $p_r(k)$ are defined in (5).

$Xflow_l$ denotes the gain of crossing nets decreased on the left edge of *bin(i, j)* when moving C_k from *bin(i, j)* to *bin(i-1, j)*. If it is positive, moving C_k to the left will lead crossing nets on the edge decreased. It means a decrease in overflow on the edge. $Xflow_r$ denotes the gain when moving right. And it is in the symmetric form for the vertical flow tendency. These parameters decide the move tendency of cells.

Moreover, the flow tendency of cells could be regarded as the net-cuts crossing the global edges. Generally speaking, reducing in net-cuts is consistent with reducing in wirelength, so moving cells according to the flow tendency will lead a decrease in wirelength.

3.3 Congestion Reducing Based on Integer Programming

For a congest bin, cells inside should move out to reduce the nets inside and achieve more free space for routing. If *bin(i, j)* is vertical-congest, cells in it should move along the horizontal direction as shown in Fig.5:

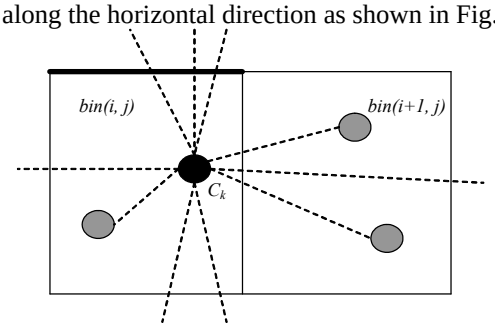

Fig.5 Move Cell to Reduce Congestion

When C_k is moved into *bin(i+1, j)*, the vertical route in *bin(i, j)* will decrease by:

$$gain_{\,v}(c_k) = \sum_{net_{kt}} p_t(k) + \sum_{net_{kb}} p_b(k) \qquad (10)$$

where net_{kt} are nets connected to C_k and their centers are in the top bins. net_{kb} are nets whose centers are in the bottom bins. $p_t(k)$ and $p_b(k)$ are the possibility of crossing the top and bottom edges of *bin(i, j)* estimated by the route model. Whether C_k should move to the left or the right bin is according to $Xflow_l(C_k)$ and $Xflow_r(C_k)$. It assures that the horizontal congestion will not increase when reducing vertical congestion. To reduce the horizontal congest is in the symmetric method.

Note that the vertical route demand in *bin(i+1, j)* will increase after C_k moves into it. This may cause unexpected congest regions. Furthermore, if *bin(i+1, j)* is already vertical-congested, moving C_k will lead a more severe congest in it. An arbitrative mechanism is needed to deal with the conflicts between the congestion reducing among multiple congest regions. An integer programming problem is constructed to resolve it.

As mentioned above, only the cells with one or more positive flow tendency could be moved. Experimental results show that the conditions are too strict. For some circuits there are few cells could move to reduce congestion. So we relax the conditions here. For a congest bin *bin(i, j)*, horizontal-movable cells should satisfy the following conditions:

$$xflow_{\,l}(c_k) > t_h \quad \text{or} \quad xflow_{\,r}(c_k) > t_h \qquad (11)$$

where t_h is a negative threshold. It means that we allow a little horizontal congest increase when reducing vertical congest. And the condition for vertical-movable cells is as follows:

$$xflow_{\,t}(c_k) > t_v \quad \text{or} \quad xflow_{\,b}(c_k) > t_v \qquad (12)$$

It is obvious that only one equation in (8) and (9) could be positive. If t_h equal 0, only one of the two inequations in (11) may be satisfied, so for a movable cell C_k, its flow orientation is certain. When we relax the conditions, both of the two inequations in (11) may be satisfied. Then we randomly prescribe the moving orientation of C_k. Later we can see that this will assure high efficiency when solving the ILP problem.

The integer linear programming problem is defined as follows:

$$\text{minimize} \quad C_{max}$$

$$\text{s.t.} \quad \begin{cases} con_{\,h}(b_{ij}) - \sum_{C_k} m_{kij}\, x_k\, gain_{\,h}(C_k) \le C_{\max} \\ con_{\,v}(b_{ij}) - \sum_{C_k} m_{kij}\, y_k\, gain_{\,v}(C_k) \le C_{\max} \end{cases}$$

$$i, j = 1, 2, \ldots, N$$

$$x_k + y_k \le 1$$

$$x_k, y_k \in \{0, 1\} \qquad k = 1, 2, \ldots, M$$

$$m_{kij} \in \{-1, 0, 1\}$$

where C_{max} is the maximum congestion degree over all the global bin.

x_k could equal 1 only when (11) is satisfied.

y_k could equal 1 only when (12) is satisfied.

$x_k + y_k \le 1$ means that C_k could not move vertically and horizontally at the same time.

m_{kij} denote which C_k should be included in the inequations. For a movable C_k in $bin(i_k, j_k)$, because its vertical and horizontal flow orientation is certain, the bin it could move into is only. So the parameter could be defined as follows:

$$m_{kij} = \begin{cases} 1 & i = i_k \text{ and } j = j_k \\ -1 & bin(i, j) \text{ is the bin } C_k \text{ moves to} \\ 0 & otherwise \end{cases} \qquad (13)$$

m_{kij} equals 1 means route demand in $bin(i_k, j_k)$ will reduce after moving C_k out. It equals -1 means route demand in the destination bin will increase for the same amount. And the perturbation to the route probability in other bins is ignored. Experimental results show that it slightly affects the final results but considerably saves the running time.

Then for some bin, the number of its related cells is limited. Actually, it is only the movable cells inside the bin and in the four adjacent bins that may be included in the inequations. The ILP problem can be optimally solved. The problem solution determines the total number and destination bins of moved cells. A post processing is then carried out to place the cells and resolve overlap inside each bin.

3.4 Post Processing

After the congestion reducing process, some cells are redistributed among different global bins. These cells should be placed without overlap. An efficient algorithm is needed to adjust the positions of cells with the minimal perturbation to the initial placement. We use the W-ECOP algorithm [5] here to accomplish the process. For a cell

need to be placed, it is inserted into the adjacent row and an optimal scheme to rearrange the cells in the row is found. If free space in the row can not accept the cell, a shifting path searching process is carried on to assure cells restrict in their neighboring area so that the performance of the circuit will be preserved.

4. Experimental Results

The algorithm has been implemented in C. All the experiments were done on a Sun E450 workstation with 4GB memory. To show the effectiveness and utility of our algorithm, a part of the experimental circuits are chosen from IBM-PLACE benchmark [13] placed with a wirelength-driven placer, Dragon [14]. Other set of circuits are from industry (Ultima Company). We compare the global routing results (overflow and wirelength) for the design before and after incremental placement.

Table 3 shows the results on the circuits from industry. As one can see, the congestion reducing approach considerably reduces the total overflow. It has a 50 percent cut down in average. And the total wirelength increased less than 0.1 percent compared to the initial placement. Some circuits even have a decrease in wrelength after placement. This indicates that the wirelength is not sacrificed much due to the reducing of congestion. It is owed to the wirelength optimization approach in W-ECOP algorithm.

Table 4 shows the results on the IBM-PLACE benchmarks placed with Dragon. Dragon has done wirelength and routability optimization by combining powerful hypergraph partitioning package with simulated annealing technique [14]. From the results we can see that the total overflow can be reduced continually through our method. And the optimization in wirelength is preserved. The algorithm is much faster on the benchmarks than on the industry circuits. The short amount of running time shows that our method can scale well for large circuits.

5. Conclusion

A new incremental placement algorithm for congestion alleviation is presented in this paper. The proposed algorithm automatically evaluates the routing congestion of a detailed placement with a fast and accurate routing estimation model. Congestion areas on the chip are relieved through cell moving. An integer linear programming (ILP) problem is formulated to resolve conflicts among multiple congest areas and avoid causing unexpected congest areas. After that an efficient algorithm for resolving overlap is used to ensure perturbing the initial placement the least. Experimental results demonstrate the effectiveness of the new approach.

Table 3 Experimental results on industry circuits

Circuits	#Cells	Grids	V/H Cap*	Overflow			Wirelength			Runtime (s)
				BIP**	AIP**	Dec. %	W(mm)	W'(mm)	Inc. %	
Cnt100	760	5 x 5	15/15	18	12	**-33.33**	46376	46961	+1.26	2.01
Cnt1000	8150	20 x 20	18/18	54	13	**-75.93**	1684936	1688686	+0.22	19.53
M32_my	7150	20 x 20	22/22	252	132	**-47.62**	962017	969448	+0.77	2.72
Gfsm300	4154	20 x 20	15/15	52	26	**-50.0**	727760	725522	-0.31	4.71
Sony_1	24847	40 x 40	57/57	442	419	**-5.2**	9940513	9928071	-0.13	193.09
Toshiba	16444	30 x 30	61/61	387	227	**-41.34**	4264968	4259167	-0.14	63.87

*V/H Cap: The vertical/horizontal capacity of each grid. **BIP/AIP: overflow before and after incremental placement

Table 4 Experimental results compared with Dragon on IBM-PLACE benchmarks

Circuits	#Cells	Grids	V/H Cap	Overflow			Wirelength			Runtime (s)
				BIP	AIP	Dec. %	W(mm)	W'(mm)	Inc. %	
Ibm01	12,036	20 x 20	27/45	156	132	**-15.38**	5171658	5118393	-1.03	2.94
Ibm02	19,062	25 x 25	54/85	597	552	**-7.53**	16403624	16422055	+0.11	12.95
Ibm03	21,924	30 x 30	36/49	413	363	**-12.10**	14193033	14210763	+0.12	7.08
Ibm04	26,346	35 x 35	38/52	337	275	**-18.40**	16057004	16093988	+0.23	6.91
Ibm05	28,146	40 x 40	67/110	546	415	**-23.99**	42195069	42230608	+0.08	13.49

References

[1] A. E. Dunlop and B.W. Kernighan, “A procedure for placement of standard cell VLSI circuits,” IEEE Trans. Computer Aided Design, vol. 4, pp. 92–98, Jan. 1985.

[2] H. Eisenmann and F. M. Johannes, “Generic global placement and floorplanning,” in Proc. Design Automation Conf., 1998, pp. 269–274.

[3] Saab et.al, “A fast clustering-based Min-cut placement algorithm with simulated-annealing performance,” VLSI Design: Int. J. Custom-Chip Design, Simulation, Testing, vol. 5, no. 1, pp. 37–48, 1996.

[4] G. Meixner and U. Lauther, “Congestion-driven placement using a new multi-partitioning heuristic,” in Proc. Int. Conf. Computer-Aided Design, Nov. 1990, pp. 332–335.

[5] Zhuoyuan Li, Weimin Wu, Xianlong Hong, Jun Gu, “Incremental placement algorithm for standard-cell layout”, Circuits and Systems, 2002 IEEE International Symposium on , Volume: 2 , 2002, Page(s): 883 -886.

[6] Wenting Hou, Hong Yu, Xianlong Hong, Yici Cai, Weimin Wu, Jun Gu, Kao W.H., "A new congestion-driven placement algorithm based on cell inflation", Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific , 2001, Page(s): 605 -608

[7] Xiaojian Yang; Kastner, R.; Sarrafzadeh, M., "Congestion reduction during placement based on integer programming", Computer Aided Design, 2001. ICCAD 2001. IEEE/ACM International Conference on , 2001, Page(s): 573 -576

[8] Maogang Wang; Xiaojian Yang; Sarrafzadeh, M., "Congestion minimization during placement", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Volume: 19 Issue: 10 , Oct. 2000, Page(s): 1140 -1148

[9] Maogang Wang; Sarrafzadeh, M., "Modeling and minimization of routing congestion", Design Automation Conference, 2000. Proceedings of the ASP-DAC 2000. Asia and South Pacific , 2000, Page(s): 185 -190

[10] Maogang Wang; Sarrafzadeh, M., "On the Behavior of Congestion Minimization During Placement", International Symposium on Physical Design, April 1990, Page(s): 145-150

[11] O. Coudert, J. Cong, S. Malik, M. Sarrafzadeh, "Incremental CAD", Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on , 2000, Page(s): 236 -243

[12] J. Cong and M. Sarrafzadeh, "Incremental Physical Design", Proc. International Symposium on Physical Design, San Diego, California, April 2000, Page(s): 84-92

[13] http://er.cs.ucla.edu/benchmarks/ibm-place/

[14] http://er.cs.ucla.edu/Dragon/