Accurate and Efficient Flow based Congestion Estimation in Floorplanning

Zion Cien Shen
Electrical and Computer Engineering
Iowa State University
Ames, Iowa, 50010
Tel: 515-294-7706
Fax: 515-294-4657
e-mail: zionshen@iastate.edu

Chris C. N. Chu
Electrical and Computer Engineering
Iowa State University
Ames, Iowa, 50010
Tel: 515-294-3490
Fax: 515-294-4657
e-mail: cnchu@iastate.edu

ABSTRACT

Congestion has been a topic of great importance in the floorplanning of deep-submicron 0design. In this paper, we design an accurate and efficient congestion estimation model by performing global routing. We interpret the global routing problem as a flow problem of several commodities and relax the integral flow constraints. The objective of resulting fractional flow problem is to minimize the *maximum congestion* over all edges in the inner dual graph [13]. The underlying routing graph for each commodity is derived by assigning directions to the inner dual graph edges. We design an efficient two-phase algorithm to solve this fractional flow problem. The first phase is denoted as *Incoming Flow Balancing* (IFB) by which a good initial solution is derived. The second phase is called *Stepwise Flow Refinement* (SFR) by which the maximum congestion of the solution in first phase is iteratively reduced to its *optimal* value. In addition, a valid global routing solution can be obtained by applying a simple rounding procedure on the fractional flow solution. The maximum congestion after rounding is only increased by 2.82% on average according to our experimental results, which justifies the use of fractional flow to estimate the routing congestion. Finally, we demonstrate our model by integrating it into a simulated annealing (SA) based floorplanner, where we use the maximum congestion as part of the cost of SA. The experimental results show that, on average, our congestion-driven floorplanner can generate a much less congested floorplan (-36.44%) with a slight sacrifice in area (+1.30%) and wirelength (+2.64%). The runtime of the whole SA process is only increased moderately (+270%).

1. INTRODUCTION

Floorplan design is to produce a chip-level plan of a set of circuit modules by determining their positions and shapes on the chip. It is the first stage of the physical design process. Hence, it has significant effects on overall circuit quality. Traditionally, in floorplanning stage, the major objective is to minimize area and total wirelength. The routability and congestion issues are not considered until global routing. However, due to the continued scaling of VLSI technology, the design of chip-level interconnect has become increasingly complicated [1]. Traditional floorplanners will produce floorplans with congested routing regions that are difficult to eliminate in later stages. Therefore, it is necessary to pay attention to the congestion optimization at floorplanning stage in order to realize the single-pass design methodology.

1.1 Previous Work

In the past few years, several works have been proposed to address the congestion issue in floorplan design. Until recently, all previous congestion models in literature divide the whole chip area into tiles [2, 3, 4, 5, 6]. The number of wires crossing a tile boundary is estimated and is used as a measure of congestion. In other words, the underlying routing graph is a grid graph [1], in which each vertex corresponds to a tile and each edge corresponds to a tile boundary. There is a tradeoff between the accuracy of congestion estimation and the cost of computation. If the number of tiles is small, the congestion estimation will be inaccurate. If the number of tiles is large, the computation will be expensive.

To estimate the congestion of each edge in the routing graph, previous approaches can be divided into two categories. The first category performs global routing on the grid graph [2]. Because the congestion estimation is performed inside the inner loop of the floorplanner, it is important to reduce the runtime of global routing. Thus, they restrict the routing geometry to L-shaped and Z-shaped. As a result, the congestion estimation may not correlate well with the real congestion. In addition, since all nets are routed one by one during global routing, even with restricted routing geometry, the computation is still very expensive.

The second category applies a probabilistic map to estimate the probability of a net crossing each boundary [3, 4, 5, 6]. The congestion of a boundary is the summation over all nets of the probability on that boundary. The previous publications differ mainly in their probabilistic maps for a net and in the way they handle blockages. This approach is more efficient than the restrictive global routing approach in the first category. Therefore, this idea is also commonly used in placement stage to estimate congestion [7, 8, 9, 10, 11]. Notice that the probability distribution of a net is determined independent of other nets. Whereas, a realistic global router routes a net based on current routing congestion information. Thus, the congestion estimated by the probabilistic approach can be very different from that by a global router. For example, in Figure 1, we have two 2-pin nets $\{A, B\}$ and $\{C, D\}$. Their bounding boxes overlap in region II. By the probability approach, we will reach the conclusion that routing region II is more congested than regions I and III. However a global router can avoid congestion by routing net $\{A, B\}$ in region I and $\{C, D\}$ in region III. The resulting congestion in II can even be less than that in I and III.

Recently, Lai et al. [12] proposed a novel approach which is very different from all previous approaches. For a floorplan of n modules, $2n$ regions are defined according to the structure of the floorplan. Each region contains several adjacent modules. The congestion for a region is evaluated as the wire density passing through the boundary of the region (i.e., number of wires connecting modules inside the region to those outside divided by the length of the region boundary). An $O(n \log n)$ time algorithm based on least common ancestor computation is presented to evaluate the congestion of all regions. This approach is very efficient. However, since most regions are quite large and only a single number is provided for each region, only coarse congestion information can be provided.

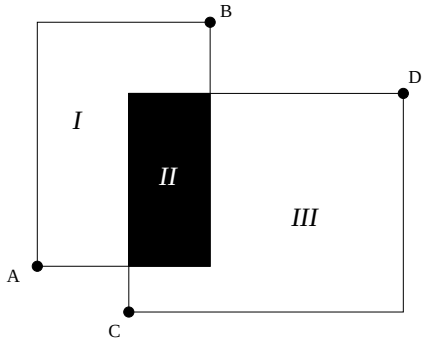

Figure 1: Congestion estimation by the probability approach.

1.2 Our Contributions

In this paper, we present a new congestion estimation model which is efficient and accurate. The basic idea is to perform global routing by a flow based approach to minimize the maximum congestion over channel segments. A *channel segment* is a segment of a channel shared by two adjacent rooms in a floorplan. If we represent each room by a vertex and connect each pair of adjacent rooms by an edge, the resulting graph is called an inner dual graph [13]. (See Figure 2 for an example.) Note that each edge in the inner dual graph corresponds to one channel segment. We use the inner dual graph as the underlying topology in global routing. In floorplanning, the exact pin positions inside a module are still unknown. It is a waste of time to use a fine grid graph to estimate the routing congestion. It is enough for global routing to list out the set of rooms that a net passes through without specifying its exact route inside each room. Therefore, the inner dual graph is an ideal choice as the underlying routing topology. The size of the inner dual graph is linear to the number of modules and is typically much smaller than that of grid graph. Henece, itt is much more efficient to use.

Inner dual graph is an undirected graph. In order to avoid detour in the routing solution, for each set of nets originating from a particular module, we use a different routing graph by assigning different directions to the edges of the inner dual graph. In order to solve this problem, we interpret it as a flow problem and we relax the integral flow constraints. We design an efficient two-phase algorithm to tackle this fractional flow problem. In the first phase, we propose an Incoming Flow Balancing (IFB) technique to derive a good initial fractional routing solution. In the second phase, we present a Stepwise Flow Refinement (SFR) technique to iteratively reduce the maximum congestion of the solution in the first phase. We prove that SFR always converges to the optimal solution. Since we relax the integral flow constraints, the optimal solution by our algorithm will only be a lower bound on the maximum congestion. A valid global routing solution can be obtained by a simple rounding procedure. We show experimentally that the maximum congestion after rounding is only increased by 2.82% on average. It justifies the use of fractional flow to estimate the routing congestion.

We demonstrate our model by integrating it into a simulated annealing (SA) based floorplanner. The maximum congestion is used as part of the cost of SA. The experimental results show that, on average, our congestion-driven floorplanner can generate a much less congested floorplan (-36.44%) with a slight sacrifice in area (+1.30%) and wirelength (+2.64%). The runtime of the whole SA process is only increased moderately (+270%). The efficiency of our model is because of the use of inner dual graph, the simplicity of the two-phase algorithm, and the fact that we route a set of nets simultaneously rather than net by net.

The remainder of the paper is organized as follows. In Section 2, we will give an overview of our congestion-driven floorplanner. In Section 3, we will present the two-phase algorithm used to solve the fractional flow problem in detail. In Section 4, a rounding procedure will be presented to derive a global routing solution from the fractional flow. The experimental results will be described in Section 5. Finally, the paper will be concluded in Section 6.

2. OVERVIEW OF OUR FLOORPLANNER

We make use of Twin Binary Sequences (TBS) [14] as our floorplan representation in simulated annealing. Basically, our congestion model can be employed with any floorplan representation. In our approach, we choose TBS representation because of two reasons. First, TBS itself is a very efficient and effective floorplan representation for mosaic floorplan and can be extended to represent general floorplan. Second, the inner dual graph of a floorplan can be easily obtained in TBS floorplan realization step.

In the annealing process, we use a 3-stage SA with three different cost functions in different temperature ranges to reduce runtime. Firstly, at high temperature range, we only consider area and total wirelength in the cost, i.e.,

$$Cost = Area + \alpha \times Wirelength.$$

Then, at medium temperature range, we add an accurate, although not optimal, maximum congestion $Mcong_1$ derived by only IFB as an additional part of the cost, i.e.,

$$Cost = Area + \alpha \times Wirelength + \beta \times Mcong_1.$$

Finally, when the annealing process reaches low temperature range, we replace $Mcong_1$ with maximum congestion $Mcong_2$ derived by IFB and SFR. The cost thus becomes:

$$Cost = Area + \alpha \times Wirelength + \beta \times Mcong_2.$$

The following section will describe how to estimate the congestion of a given floorplan in detail.

3. CONGESTION ESTIMATION MODEL

In Section 3.1, we first introduce the inner dual graph and describe how to obtain the underlying routing graph from the inner dual graph. Then we illustrate the method of constructing the inner dual graph directly from TBS in Section 3.2. Based on the inner dual graph, in Section 3.3, we formulate the congestion minimization problem as a flow problem. Section 3.4 and 3.5 describe an efficient two-phase algorithm to tackle the problem formulated in Section 3.3.

3.1 Underlying Routing Graph

The exact pin positions inside each module is not given in the floorplanning stage. So it is not necessary to use a fine grid graph to estimate the routing congestion as in previous works. It is enough for global routing in floorplanning stage to list out the set of rooms that a net passes through without specifying its exact route inside each room.

Given a floorplan R, the room adjacency relationships can be described by *channel segment* which is defined as a segment of a channel shared by two adjacent rooms in the floorplan. For example, in Figure 2(a), the channel segment corresponding to the adjacent rooms C and D is highlighted. The room adjacency relationships can also be represented by the inner dual graph $G = (V, E)$ [13] where

$$V = \{v | v \text{ corresponds to a room of } R\}$$
$$E = \{\{u, v\} | u \text{ and } v \text{ are adjacent to each other in } R\}$$

See Figure 2(b) for an example. Note that there are one-to-one mappings between the rooms in R and the vertices in G, and between the channel segments in R and the edges in G. In the rest of the paper, the terms floorplan room and inner dual graph vertex, and channel segment and inner dual graph edge are used interchangeably.

The inner dual graph can be used as the underlying graph in global routing. The size of the inner dual graph is linear to the number of modules and is typically much smaller than the size of grid graph used in previous approaches. Hence, it is much more efficient to use. However, the inner dual graph is an undirected graph. If it is used directly as the underlying routing graph, the routing solution may have a lot of detour. We avoid detour by assigning directions to the edges of inner dual graph. Notice that different nets may require different direction assignments, but all nets originating from the same module share the same direction assignment. So for each set of nets originating from a specific module, we can derive a specific directed acyclic

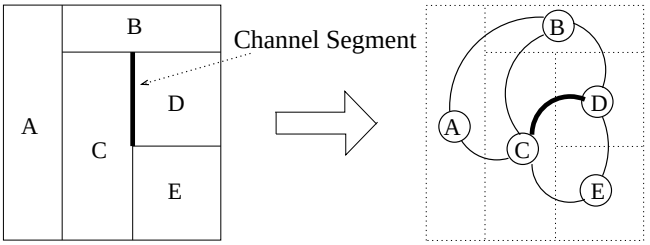

Figure 2: (a) A rectangular floorplan R. (b) Its inner dual graph G. The channel segment and the inner dual graph edge corresponding to the adjacent rooms C and D are highlighted.

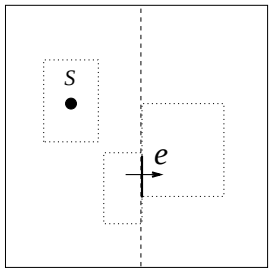

Figure 3: An illustration of routing direction assignment.

graph (DAG) G' as the routing graph according to the following rule (as illustrated in Figure 3). Consider nets originating from a source room s, and consider a channel segment e shared by a pair of adjacent rooms in floorplan R. By extending the channel segment, the floorplan region will be divided into two sides. We assign the direction of the edge e in the inner dual graph to be from the room on the same side as the center of s to the other side. See Figure 4 for an example. Notice that even with direction assignment, some detour may still occur. For example, in Figure 4, a net following the path $< D, E, C, A >$ may have detour depending on the exact pin positions inside rooms D and A. However, with direction assignment, major detour can be avoided.

We observe that the underlying routing graph for a specific commodity obtained by the assignment above is a directed acyclic graph (DAG) in most cases. However, there exists a special case as illustrated in Figure 5. When D is considered as a source room, a cyclic path $< A, B, C, E, A >$ exists in this graph. To obtain an acyclic underlying routing graph $G' = (V', E')$, we remove one edge from each cycle based on the rule described in Section 3.4.

3.2 Inner Dual Graph Construction from TBS

The inner dual graph G describes the neighborhood information between any two rectangular rooms. Given a floorplan in TBS, we can construct its inner dual graph in linear time by finding all pairs of adjacent rooms. In TBS representation, each floorplan is one-to-one mapped to a pair of twin binary trees (t_1, t_2). t_1 and t_2 are obtained

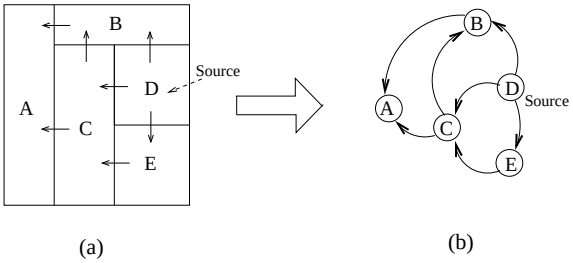

Figure 4: An example of routing direction assignment.

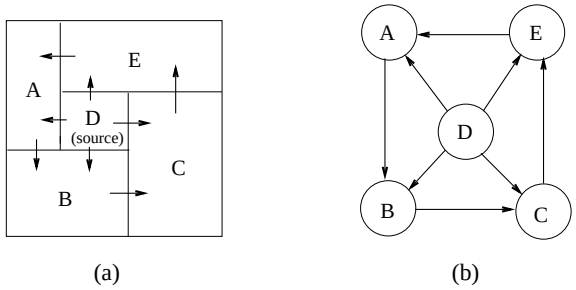

Figure 5: A special case where cycle exists after direction assignment.

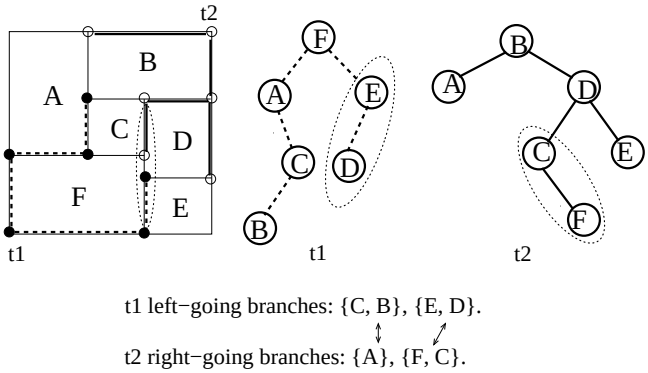

Figure 6: Determining the room neighborhood information directly from TBS.

by connecting, respectively, lower-left corners and upper-right corners of all rooms. We use the floorplan in Figure 6 as an example to illustrate how to obtain all room adjacency relationships as well as the length of each channel segment directly from TBS. We define a left-going (right-going) branch of a binary tree to be any right (left) child and all its left (right) descendents. For example, in t_1, the left-going branches are $\{C, B\}$ and $\{E, D\}$. In t_2, the right-going branches are $\{A\}$ and $\{F, C\}$. We notice that for each vertical channel (not including the boundaries), the room(s) on its right side corresponds to the room(s) in a particular left-going branch of t_1. The room(s) on its left side corresponds to the room(s) in a particular right-going branch of t_2. See the vertical channel highlighted in Figure 6 as an example. If there are m vertical channels in the floorplan, there will also be m left-going branches in t_1 and m right-going branches in t_2. The branches have already been found in the original TBS packing procedure. Thus, in order to capture all horizontal adjacency relationships as well as the length of each vertical channel segment, we only need to compare the heights of rooms in a left-going branch of t_1 with those in a corresponding right-going branch of t_2 (i.e., $\{C, B\}$ with $\{A\}$, $\{E, D\}$ with $\{F, C\}$). Similarly, we can obtain the vertical adjacency relationships and horizontal channel segment lengths by considering the horizontal channels (i.e., the right-going branches in t_1 and the left-going branches in t_2).

3.3 Problem Formulation

In our formulation, as in previous congestion estimation papers, we only handle 2-pin nets for the sake of simplicity. Notice that multi-pin nets can be easily broken down to several 2-pin nets by Minimum Spanning Tree or Rectilinear Steiner Tree techniques.

Our congestion model is meant to estimate the best maximum congestion over all possible global routing solutions. If we do not assign directions to the inner dual graph G, we can formulate the global routing problem as a flow problem with several commodities, where each commodity corresponds to a set of nets originating from a particular module. We first introduce some notations.

- N_i: the set of neighboring vertices of vertex i.
- c_i^k: the demand of vertex i for commodity k, i.e., the total

number of nets with source vertex k and sink vertex i.

- f_{ij}^k: the amount of flow from i to j for commodity k for $\{i,j\} \in E$ and $j \neq k$.
- cap_e: the capacity of channel segment e in R, i.e., the maximum number of nets that can cross it.
- $cong_e$: the congestion of channel segment e, i.e., the ratio of the number of nets crossing it to its capacity.
- $Mcong$: the maximum congestion over all channel segments of floorplan R.

Note that all flow amount f_{ij}^k should be integral in order to be a valid global routing solution. Also note that f_{ij}^k may be different from f_{ji}^k. For the capacity of channel segment, it is technology dependent. We can calculate it as follows. Let l_e be the length of the channel segment e. Let b be the sum of minimum wire width and minimum wire spacing. Then the routing capacity is calculated as $cap_e = \lfloor l_e/b \rfloor$. In general, the capacity can also be modified to model routing blockage.

The congestion of edge $e = \{i,j\}$ can be written as follows:

$$cong_e = \frac{\sum_k (f_{ij}^k + f_{ji}^k)}{cap_e}.$$

Then, the flow problem can also be formulated as the following integer linear program (ILP):

$$\begin{aligned}
\text{Minimize} \quad & Mcong \\
\text{such that} \quad & \frac{\sum_k (f_{ij}^k + f_{ji}^k)}{cap_e} \leq Mcong, \ \forall e = \{i,j\} \in E \quad (1) \\
& \sum_{j \in N_i} f_{ji}^k = \sum_{j \in N_i} f_{ij}^k + c_i^k, \ \forall i,k \in V \text{ s.t. } i \neq k \quad (2) \\
& f_{ij}^k \geq 0, \ \forall i,j,k \in V \quad (3) \\
& f_{ij}^k \in \mathbf{Z}, \ \forall i,j,k \in V \quad (4)
\end{aligned}$$

Constraint (2) is the flow conservation constraint. It specifies that for each commodity k and for each vertex $i \neq k$, the total incoming flow equals the total outgoing flow plus the demand of vertex i. Note that by summing constraint (2) over all $i \neq k$, we can derive:

$$\sum_{j \in N_k} f_{kj}^k = \sum_{i \neq k} c_i^k, \ \forall k \in V$$

This means for commodity k, the total outgoing flow from vertex k equals the total demand.

Since we restrict the flow direction for different commodity as described in Section 3.1, we need to add $O(n^2)$ constraints to the ILP formulation above. For each commodity k and each edge $\{i,j\} \in E$, if the flow direction is from i to j, we add the constraints $f_{ij}^k \geq 0$ and $f_{ji}^k = 0$.

ILP is known to be NP-complete. To tackle this problem, we first relax the integral flow constraint (4). Notice that the resulting problem is similar to the classical maximum concurrent flow problem [15]. However, in our problem, the flow direction on each edge may differ for different commodities. Our problem can be solved by any LP solver. However, it is too time consuming to be applied in the inner loop of the floorplanning process. Instead, we propose an efficient two-phase algorithm to derive the optimal fractional flow solution. The algorithm will be explained in detail in the following two subsections.

3.4 Incoming Flow Balancing (IFB) Phase

In this Section, we present an Incoming Flow Balancing (IFB) technique to derive a good fractional flow solution. This solution is included into the cost of the second stage of SA and is also used as an initial solution of SFR technique described in Section 3.5.

We construct the flow solution by iteratively deriving the flow of each commodity one by one based on current congestion information. At the beginning, we set the flow amount and congestion on each edge for each commodity to 0. When considering commodity k, we obtain the underlying routing graph $G' = (V', E')$ by assigning the directions to each edge of the inner dual graph G. If cycle occurs, we remove the most congested edge in the cycle according to the current congestion information. Then we consider vertices in reverse topological order[1] [16] of graph G'. For each vertex i, in order to minimize the maximum congestion, we balance incoming flow to make the congestion of incoming edges as even as possible. Let d_{in} be the number of incoming edges for vertex i. Let fin_j be the flow amount of commodity k, $congin_j$ be the current congestion excluding commodity k, and $capin_j$ be the capacity of the j-th incoming edge ($1 \leq j \leq d_{in}$). Our goal to minimize the maximum congestion over all incoming flow can be written as follows:

$$\text{Minimize} \quad \left\{ \max_{1 \leq j \leq d_{in}} \left\{ congin_j + \frac{fin_j}{capin_j} \right\} \right\}.$$

Since we consider the vertices in reverse topological order, all outgoing flow of i has already been determined. Let f_{out} be the total outgoing flow amount. Then the flow conservation constraint in equation (2) can be rewritten as follows:

$$\sum_{j=1}^{d_{in}} fin_j = c_i^k + f_{out} \quad (5)$$

This problem can be easily solved by adding flow to the least congested incoming edge(s) until its congestion matches that of the next least congested edge. Note that there may be more than one edges with the least congestion. In that case, we add flow to them such that they still have the same congestion. We keep on adding flow until equation (5) is satisfied. See an example in Figure 7. For commodity k, The total outgoing flow amount is 6. The demand for vertex i is 2. So the total incoming flow amount should be $6 + 2 = 8$. We assign $fin_1 = 0, fin_2 = 2, fin_3 = 6$ to edges 1, 2, and 3, respectively. The maximum incoming flow congestion is thus 0.6.

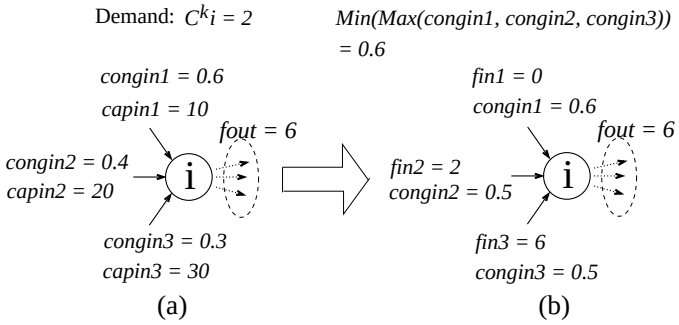

Figure 7: Illustration of Incoming Flow Balancing technique. (a) Flow distribution before assigning incoming flow. (b) Flow distribution after assigning incoming flow.

The procedure of routing all commodities once is called a *pass*. In IFB phase, we perform several passes until the maximum congestion converges. Notice that in pass $i \geq 2$, for commodity k, we first remove all its flow in pass $i-1$ and update the congestion. Then we balance the flow of incoming edges according to the updated congestion. Since our algorithm is very greedy, the maximum congestion will converge in 2 to 4 passes in practice. An example of the convergence of the MCNC benchmark $apte$ circuit is shown in Figure 8. The overall flow of this phase is summarized in the IFB Algorithm below.

3.5 Stepwise Flow Refinement (SFR) Phase

Since IFB phase can only achieve local incoming flow balance at each step, we still need an additional phase to obtain global solu-

[1] A reverse topological order of a directed acyclic graph (DAG) is a linear ordering of all its vertices such that if it contains an edge (u, v), then u appears after v in the ordering. For instance, in Figure 4(b), the corresponding reverse topological order is A, B, C, E, D.

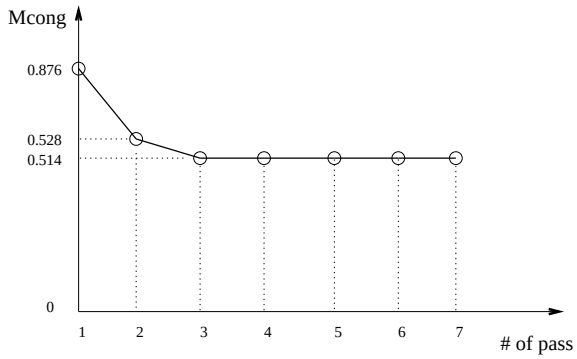

Figure 8: The convergence of maximum congestion in IFB phase for circuit $apte$.

IFB Algorithm:
Input: Inner dual graph $G = (V, E)$
Output: $cong_e$ and f_{ij}^k on each vertex k and edge $e = \{i, j\}$
Initialize $cong_e$ and f_{ij}^k to 0 for all k and $e = \{i, j\}$;
While $Mcong$ is not converged **do**
  For each commodity k **do**
    Assign flow directions to all edges to obtain DAG G';
    If it is not the first pass,
      remove f_{ij}^k, and update $cong_e$ for each edge;
    Do a reverse topological ordering on DAG G';
    For each vertex in a reverse topological order **do**
      Assign incoming flow in a balanced manner;

tion by stepwise refining the flow solution given by IFB phase. An *iteration* in SFR phase is illustrated in Figure 9.

First, we pick an edge $e = \{i, j\}$ with maximum congestion $Mcong$. Second, we pick a commodity k which contributes more than $\gamma\%$ of total flow amount on edge e (i.e., $f_{ij}^k > Mcong \cdot cap_e \cdot \gamma\%$). Third, based on the DAG G' of commodity k, we find the set of vertices V_{out} which is reachable from j. V_{out} includes j. Similarly, we find the set of vertices V_{in} which can reach i. V_{in} includes i. Then, by applying breadth first search (BFS), we find a simple path r_1 which links a vertex $p \in V_{in}$ and another vertex $q \in V_{out}$. In the meantime, we require that r_1 should not pass through edge e and the maximum congestion over the edges of r_1 should be at least ϵ less than $Mcong$. At the same time, we derive another path r_2 which connects p and q by passing through edge e. Finally, we move δf amount of flow from r_2 to r_1 in a way that the maximum congestion over the edges of r_1 and r_2 is minimized. δf should also satisfy an additional constraint that the total incoming flow amount for each vertex on r_2 after moving δf flow amount should not be less than its demand of commodity k. Finally, we update the congestion and flow amount on edges of r_1 and r_2. We keep applying the SFR technique on the edge with maximum congestion iteratively until it is not able to find r_1 and r_2 for the current most congested edge after all commodities are tried. In practice, we could speed up this process by tuning the parameter γ and ϵ. A convergence of $Mcong$ in SFR phase is shown in Figure 10, where the test circuit is $apte$, and γ and ϵ are set to 10 and 0.001, respectively. We cannot find routes r_1 and r_2 to further improve $Mcong$ after 13 iterations. SRF Algorithm gives the overall flow of the procedure above.

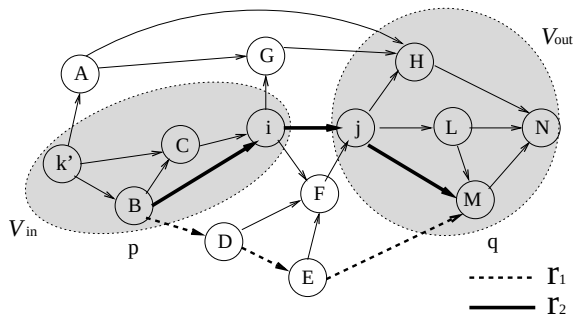

Figure 9: SFR approach to globally optimize the maximum congestion. In this example, $e = \{i, j\}$ is with $Mcong$, $V_{out} = \{j, H, L, M, N\}, V_{in} = \{i, C, B, k\}$. After one iteration, we derive $r_1 =< B, D, E, M >$ and $r_2 =< B, i, j, M >$.

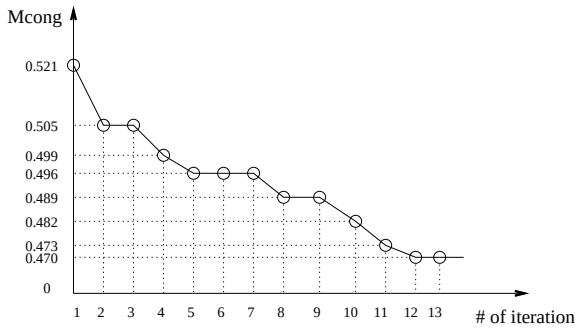

Figure 10: The convergence of $Mcong$ of $apte$ in SFR phase, where $\gamma = 10$, $\epsilon = 0.001$.

LEMMA 1. *If $\gamma = \epsilon = 0$, the SFR algorithm always converges to the optimal solution.*

PROOF. Note that if the current flow solution is not optimal, SFR can always find two routes r_1 and r_2 to reduce the maximum congestion. Since the maximum congestion is bounded below, the SFR algorithm always converges to the optimal solution. □

SFR Algorithm:
Input: Inner dual graph G with initial solution obtained from IFB algorithm.
Output: Minimized maximum congestion $Mcong$.
Do
  Pick an edge $e = \{i, j\}$ with maximum congestion;
  For each k with f_{ij}^k (or f_{ji}^k) $> Mcong \cdot cap_e \cdot \gamma\%$ **do**
    Find V_{in} and V_{out};
    Find r_1 and r_2;
    Move δf from r_2 to r_1 and update the flow amount and congestion of edges on r_1 and r_2;
While edge e can find r_1 and r_2

4. GLOBAL ROUTING SOLUTION GENERATION

Based upon the final floorplan solution at the end of simulated annealing, we obtain a global routing solution by a simple rounding technique applied to the fractional flow solution. For each commodity k, we round the incoming flow of vertices in reverse topological order. In the process, we maintain the flow conservation constraint in equation (2) by adjusting the flow of the least congested incoming edge. Once the rounding process for all commodities is finished, we update the congestion for each edge. Then, we apply SFR to optimize the global routing solution. It is important to stress that this time we only allow to move an integral amount of flow, namely δf, from one congested route to another less congested route.

5. EXPERIMENTAL RESULTS

We implement the algorithm in the C programming language and test five MCNC benchmarks on a Sun4u machine with 8 GB memory and 750MHz Sparcv9 processor. The parameters of those benchmarks are listed in Table 2. The modules in the benchmark are soft modules with aspect ratio 0.5 to 2. To calculate the capacity of each channel segment, we assume the sum of minimum wire spacing and

MCNC benchmark	$Area\ (mm^2)$		$Wirelength\ (mm)$		$FMcong \rightarrow IMcong$		$Runtime\ (s)$	
	F1	F2	F1	F2	F1	F2	F1	F2
apte	48.227	47.898	132.67	134.33	0.607 → 0.611	0.371 → 0.373	2.10	4.69
xerox	20.243	19.864	147.38	152.01	1.059 → 1.070	0.938 → 0.945	2.58	4.47
hp	9.397	10.019	44.93	48.91	1.931 → 2.014	0.803 → 0.826	1.74	3.73
ami33a	12.468	12.636	64.88	64.91	1.453 → 1.502	0.980 → 0.993	4.21	13.63
ami49a	39.748	40.221	302.56	294.53	2.570 → 2.738	1.590 → 1.699	8.14	75.04
average (F2-F1)/F1	+1.30%		+2.64%		-35.88% → -36.44%		+270%	

Table 1: Experimental results for MCNC benchmarks.

minimum wire width to be 6λ. Since our congestion model is based upon 2-pin net, we decompose each multi-pin net into a set of 2-pin nets. In addition, we randomly choose one pin as the source since the benchmarks are lack of signal direction information. In the case that one pin is located along the chip boundary, we assign this pin to its corresponding boundary room. Thus, each net starts from one room and ends at another room.

Circuit	modules	nets	2-pin nets
apte	8	97	172
xerox	10	203	455
hp	11	83	226
ami33a	33	123	363
ami49a	49	408	545

Table 2: MCNC benchmark.

In order to test the effectiveness and efficiency of our algorithm, we compare two floorplanners: F1, without congestion optimization; F2, with congestion optimization. In F1, a single stage simulated annealing is used to search for a floorplan aiming at minimizing total area and wirelength only. The maximum congestion is obtained by applying IFB and SFR to the final floorplan. In F2, we employ aforementioned 3-stage simulated annealing to obtain a floorplan with minimized weighted sum of area, wirelength, and maximum congestion. The weights are set such that the costs of area, wirelength, and congestion are approximately equal. The initial temperature in annealing process is set to 1.5×10^6 and drops down at a constant rate of 0.95 to 0.97 until it is below 1×10^{-10}. The number of iterations at one temperature step is 30. For each experiment, 10 runs are performed and the result of the best run is reported. The area, total wirelength, maximum congestion and total runtime are reported in Table 1. In terms of maximum congestion, $FMcong$ denotes the maximum congestion based on fractional flow and $IMcong$ denotes the maximum congestion based on integral flow after rounding. The experimental results show that the maximum congestion after applying rounding increases only by 2.82%, on average. This means our fractional flow based congestion model is fairly accurate in terms of estimation of the congestion for a floorplan.

From Table 1, we can notice that, compared to floorplanner F1, the congestion-driven floorplanner F2 can reduce the maximum congestion by 36.44% on average. More specifically, for $ami33a$, hp and $xerox$, F1 is not able to produce routable final floorplan solutions (as their maximum congestions exceed 1), whereas, F2 is able to produce routable solutions. For $apte$, both F1 and F2 generate a routable floorplan solution. However, F2 can reduce maximum congestion by almost 40% as compared to F1. For $ami49a$, both F1 and F2 are not able to produce a routable floorplan. Nevertheless, since the maximum congestion by F2 is much less than F1, the floorplan by F2 is more likely to be successfully routed if more detour is allowed. Hence, F2 can reduce the overall congestion and improve routability of a circuit significantly in floorplanning stage with slight increase of area ($+1.30\%$), total wirelength ($+2.64\%$), and runtime ($+270\%$).

6. CONCLUSION AND DISCUSSION

In this paper, we presented a flow based congestion estimation model for estimating the routing congestion in floorplanning level. This model is based on the inner dual graph. The two-phase algorithm used in this model is optimal and efficient in estimating the maximum congestion of a floorplan. The experimental results show that the maximum congestion can be better optimized by incorporating this congestion model into floorplanner with slight sacrifice on area and wirelength. In the future, we plan to extend our algorithm to handle timing driven floorplanning and noise-aware floorplanning.

7. REFERENCES

[1] N. Sherwani. Algorithms for VLSI Physical Design Automation, 3rd Edition. Published by Kluwer Academic Publishers, 1999.

[2] H. M. Chen, H. Zhou, F. Y. Young, D. F. Wong, H. H. Yang, and N. Sherwani. Integrated floorplanning and interconnect planning. *Intl. Conf. on CAD*, pages 354–357,1999.

[3] A. Ranjan, K. Bazargan and M. Sarrafzadeh. Fast hierarchical floorplanning with congestion and timing control. *Intl. Conf. of Computer Design*, pages 357–362, 2000.

[4] C. K. K. P. Sarkar, V. Sundararaman. Routability-driven repeater block planning for interconnect-centric floorplanning. *Intl. Symp. Physical Design*, page 186–191. 2000.

[5] C. W. Sham, W. C. Wong and F. Y. Young. Congestion Estimation with buffer planning in floorplan design. *Design, Automation and Test in Europe Conf. and Exhibition*, pages 696–701, 2002.

[6] J. Lou , S. Krishnamoorthy and H. S. Sheng. Estimating routing congestion using probabilistic analysis. *Intl. Sym. on Physical Design, ACM*, pages 112–117, 2001.

[7] M. Wang and M. Sarrafzadeh. On the behavior of congestion minimization during placement. *Intl. ASIC Conf. and Exhibit*, pages 145–150, 1999.

[8] C. L. E. Cheng. RISA: Accurate and efficient placement routability modeling. *Intl. Conf. Computer-Aided Design*, pages 690–697, 1994.

[9] 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. *Conf. Asia South Pacific Design Automation*, pages 605–608, 2001.

[10] X. Yang, R. Kastner, and M. Sarrafzadeh. Congestion reduction during top-down placement. *Int. Conf. Computer-Aided Design*, pages 573–576, 2001.

[11] Ulrich Brenner and André Rohe An efficient congestion-driven placement framework. *IEEE Transaction of Computer-Aided Design*, Volume 22, pages 387–394, 2003.

[12] S. T. W. Lai, F. Y. Young and C. N. Chu. A new and efficient congestion evaluation model in floorplanning: Wire density control with Twin Binary Trees. *Design, Automation and Test in Europe Conf. and Exhibition*, pages 856–861, 2003.

[13] S. M. Sait and H. Youssef. VLSI Physical Design Automation, page 117. Published by IEEE Press, 1995.

[14] F. Y. Young, C. N. Chu and Z. C. Shen. Twin binary sequences: A non-redundant representation for general non-slicing floorplan. *Intl. Sym. on Physical Design, ACM*, pages 457–469, 2002.

[15] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *Journal of ACM*, page 318–334, 1990.

[16] T. H. Cormen, C. E. Leiserson, R. L, Rivest and C. Stein. Introduction to Algorithms, 2nd Edition. Published by MIT Press, 2001.