Performance Driven Global Routing Through Gradual Refinement *

Jiang Hu
IBM Microelectronics
Email: jianghu@us.ibm.com

Sachin S. Sapatnekar
Department of ECE, University of Minnesota
Email: sachin@mail.ece.umn.edu

Abstract

We propose a method for VLSI interconnect global routing that can optimize routing congestion, delay and number of bends, which are often competing objectives. Routing flexibilities under timing constraints are obtained and exploited to reduce congestion subject to timing constraints. The wire routes are determined through gradual refinement according to probabilistic estimation on congestions so that the congestion is minimized while the number of bends on wires are limited. The experiments on both random generated circuits and benchmark circuits confirm the effectiveness of this method.

1 Introduction

Global routing is an important stage in VLSI physical design, in which a given set of global nets is routed coarsely, in an area that is conceptually divided into small regions called routing cells. For each net, a routing tree is specified only in terms of the cells through which it passes. For a boundary between two neighboring cells, the number of available routing tracks across it, called its supply, is limited. One fundamental goal of global routing is to minimize the congestion so that the number of nets across each boundary does not exceed its supply. Since minimizing congestion is very hard to achieve and is essential for global routing, it has long been a focus of research in global routing and results in various methods including sequential approach [1], rip-up-and-reroute technique [2], hierarchical method [3] and multicommodity flow based router [4]. When interconnect becomes a performance bottleneck in deep submicron technology, merely minimizing congestion is not enough. In later works [5, 6], interconnect delays are explicitly considered during global routing. Besides congestion and timing, the number of bends for each wire needs to be limited, since wire bend usually implies a switching of layers, which involves a via resistance that adds to the delay and reduce reliability, and will consume more wiring space. In [3], a hierarchical global routing algorithm is proposed to control to number of vias for each wire.

In global routing, congestion, delay constraints and control of the number of vias are often competing objectives. In order to avoid congestion, some wires must make detours, and the signal delay will consequently suffer. Controlling the number of vias will reduce the capability of a wire to avoid congestion, and a large number of vias will also affect the delay performance. Our work is an effort to minimize the congestion while satisfying timing constraints and limiting the number of vias for each wire in global routing. Similar to the work of [6], we obtain routing topology flexibilities under bounded delays through deferred decision making and trade them into congestion reduction under timing constraints. However, our tradeoff method is a probability-based gradual refinement which is different from [6]. Moreover, we integrate restrictions on wire routes with the refinement so that the number of bends on wires can be bounded. Although our objective is complex, our method is simple and the experimental results on both random and benchmark circuits confirm that it is effective in achieving all three objectives simultaneously.

*This work is supported in part by the NSF under contract CCR-9800992 and the SRC under contract 98-DJ-609.

2 Definitions and Problem Formulation

We are given a set of nets $\mathcal{N} = \{N^1, N^2, ...\}$, with each net N^i being defined by a source node v_0^i and a set of sink nodes $V_{sink}^i = \{v_1^i, v_2^i, ...v_p^i\}$. A routing problem for a net N is to find a set of Steiner nodes $V_{Steiner} = \{v_{p+1}, v_{p+2}, ...v_{p+q}\}$ and a set of edges $E = \{e_1, e_2, ...e_{p+q}\}$ to construct a tree $T(V, E)$, where $V = v_0 \cup V_{sink} \cup V_{Steiner}$, such that E spans all of the nodes in V. The location for a node v_j is specified by its coordinates x_j and y_j, and an edge in E is uniquely identified by the node pair (v_j, v_k), the notation e_{jk} or e_k interchangeably. We assume v_j is the upstream end of this edge. The edge length l_{jk} is given by the Manhattan distance between the two nodes, which is $|x_j - x_k| + |y_j - y_k|$. In order to make our presentation clearer, we define a *backbone node* to be the source node, or a sink node, or a Steiner node with degree greater than 2 in a routing tree. We also define a *backbone wire* to be a set of consecutively adjoined edges $\{(v, u_1), (u_1, u_2)...(u_m, w)\}$, where $v, w \in V$ are backbone nodes and none of $\{u_1, u_2, ...u_m\} \in V$ is a backbone node.

As in conventional global routing, we tessellate the entire routing region for $\mathcal{N}$ into an array of uniform rectangular cells. We represent this tessellation as a graph called the grid graph $G(V_G, E_G)$, where $V_G = \{g_1, g_2, ...\}$ corresponds to the set of grid cells, and a grid edge $b_{ij} = (g_i, g_j) \in E_G$ corresponds to the boundary between two adjacent grid cells $g_i, g_j \in V_G$. In this work, we also use $g(r, c)$ to represent a grid cell at row r and column c. There are a limited number of routing tracks across any grid edge, b, called the *supply* of the grid edge and expressed as $s(b)$. During the routing, the number of tracks occupied by wires across a grid edge b is designated as the *demand*, $d(b)$. The *overflow* $f_{ov}(b)$ at grid edge b is defined by $f_{ov}(b) = \max(d(b) - s(b), 0)$. The *demand density* for a grid edge b is defined as $\mathcal{D}(b) = \frac{d(b)}{s(b)}$. We use the maximum demand density $\mathcal{D}_{max} = \max_{b \in E_G}\{\mathcal{D}(b)\}$ and total overflow $\mathcal{F}_{ov} = \sum_{\forall b \in E_G} f_{ov}(b)$ to evaluate the congestions in the final results. In this work, we use the π RC model for wires, RC switch model for drivers and Elmore delay model for delay calculation.

For a given set of nets $\mathcal{N}$ and a grid graph G over the area of $\mathcal{N}$, our objective is to construct routing trees T^i for every $N^i \in \mathcal{N}$, such that the delay at every sink meets its given timing constraint, number of bends on each backbone wire is no greater than 5 and the congestion is minimized in terms of $\mathcal{D}_{max}$ and $\mathcal{F}_{ov}$.

0-7695-1200-3/01$10.00 © 2001 IEEE

3 Routing Flexibilities under Timing Constraints

If the timining constraint is not over-tight, there are usually multiple tree topologies that can meet the timing constraint. In [6, 7], the concepts of soft edge and slideable Steiner node(SSN) are proposed to represent certain timing-constrained routing flexibilities. In this work, we will employ both concepts together with the concept of Z-edge and edge elongation. When the number of bends along a route connecting two nodes is restricted to be no greater than two and its path length to be the Manhattan distance between the two nodes, this route can only be straight, L-shaped or Z-shaped. A *Z-edge* is an edge that can take only such a route. Even though the routing flexibility from a Z-edge is less than that of a soft edge, this flexibility can preserve timing performance with bounded number of bends. Edge elongation implies that an edge can be stretched in its length as long as no timing violation incurred so that this edge can have more flexibilities on the routes it may take.

4 Approximated Congestion Estimation

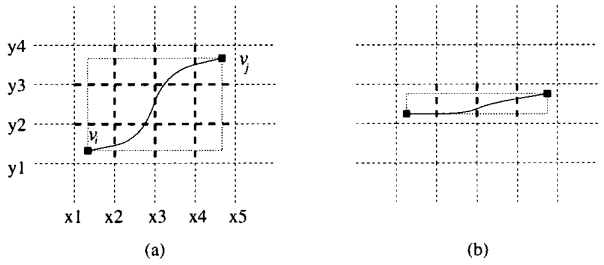

Figure 1: Examples of primitive demand. (a) each grid edge corresponds to a horizontal(vertical) thickened boundary segment has primitive demand of $\frac{1}{4}(\frac{1}{3})$. (b) each grid edge corresponds to a thickened boundary segment has primitive demand of 1.

In addition to the traditional congestion metrics, we use a couple of other approximate estimation methods during different phases of global routing. For a soft edge, we define a *primitive demand* to indicate the possibility that this soft edge will take a route across a grid edge. If the two ends of a soft edge (v_i, v_j) locate in grid cell $g(r_i, c_i)$ and $g(r_j, c_j)$ (assuming $r_i \leq r_j, c_i \leq c_j$), repectively, each upper horizontal grid edge of grid cell $\{g(r,c) \in V_G | r_i \leq r < r_j, c_i \leq c \leq c_j\}$ has a primitive demand $\frac{1}{|c_j - c_i + 1|}$ from this edge. Similarly, each righthand grid edge of grid cell $\{g(r,c) \in V_G | r_i \leq r \leq r_j, c_i \leq c < c_j\}$ has a primitive demand $\frac{1}{|r_j - r_i + 1|}$ from this edge. This is illustrated in the examples in Fig. 1. The total primitive demand $d_{prim}(b)$ on grid edge b is the summation of primitive demands from all soft edges whose bounding box overlapping with it.

If the edge (v_i, v_j) we are considering is a Z-edge, through an enumeration similar to [9], we can conclude that there are $Z = |r_i - r_j| + |c_i - c_j|$ possible routes for a Z-edge between two grid cell (r_i, c_i) and (r_j, c_j) in a grid graph. We assume a uniform probability distribution for these routes. i.e., every route has the same chance to be chosen in later stages. Then, we can obtain the probability that a grid edge is crossed by a Z-edge. For each vertical grid edge to the right of grid cell $g(r_i, c_k), c_i \leq c_k < c_j$, the probability is $|c_j - c_k|/Z$. The probability at other grid edge can be counted similarly. Based on these probabilities, we define the *probabilistic demand* from a Z-edge e_{ij} to a grid edge b to be the probability that Z-edge run across this grid edge, and is denoted as $d_{prob}(b, e_{ij})$.

5 Algorithm

Our strategy is to obtain the required timing performance first and then concentrate on optimizing the congestion and the number of bends while preserving the timing performance obtained. Therefore, we initially route each net individually through timing driven algorithms without considering congestion or the number of bends. We use soft edges and slideable Steiner nodes in this phase so that the routing result is composed of only backbone nodes and every backbone wire is a single edge. In the second phase, we will try to specify the details for the slideable Steiner nodes and backbone wires in an effort to minimize congestion and control the number of bends on each backbone wire. In contrast to sequential and rip-up-and-reroute methods, where nets routed earlier have no or little knowledge on routing requests from other nets, our method select wire routes in a gradual refinement manner according to more and more accurate congestion feedback.

In phase I, we route each net through the MVERT [8] algorithm using soft edges and slideable Steiner nodes so that the timing constraints can be satisfied and the resulting routing tree consists of only backbone nodes and each backbone wire is either a solid edge or a soft edge.

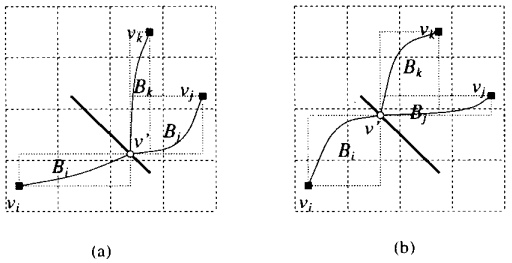

Figure 2: When an SSN slides along its locus, the bounding boxes of its incident edges change as well as the primitive demands.

After phase I, we can obtain a rough estimation of congestions through the concept of *primitive demand*. The first step in phase II is to fix the position of each SSN (Slideable Steiner Node) to reduce the peak primitive demand density. When an SSN slides along a locus of points, the lengths of its incident edges are not changed, and nor is the delay at any sink. From the example in Fig. 2, we can see that sliding an SSN may cause changes on primitive demands. Through a linear search for all the grid cells that the locus of the SSN intersects, we can find a grid cell to fix the position of the SSN such that the maximum demand density among the grid edges involved is minimized.

After fixing the SSN, we will make two sweeps of all the backbone wires in a constant order to specify their routes. Instead of specifying the complete route immediately in one step, we first only specify *one* grid cell that we force the backbone wire to to pass through. Note that neither of the end nodes of this backbone wire can be within this grid cell. We insert a pseudo node, which we call the *post node* in the backbone wire within this selected grid cell. Before choosing the grid cell for the post node, we need to choose the candidate grid cells that will be considered. A candidate grid cell is any grid cell where we can force the backbone wire passing through without causing timing violation, i.e., a backbone wire might be elongated to pass through it under timing constraint. Similarly, a formerly solid edge can also be elongated to reduce congestion.

After the post node is inserted, the former backbone wire is split into two subedges. We specify that each sub-edge can be only a Z-edge so that we can bound the number of bends for each backbone wire. Different choices on the location of the post node and the routes of its two Z-edges usually provide us with plenty of routing flexibilities under restric-

tions on timing and number of bends on wires. Moreover, by using Z-edges we can obtain a better estimation of the congestion through the probabilistic demand. We choose the post node so as to minimize the congestion cost for the two sub-edges. The congestion cost of a subedge e_j is defined as: $cost(e_j) = \sum_{\forall b \text{ intersecting } B(e_j)} \mathcal{D}(b)^2 \times d_{prob}(b, e_j)$, where $B(e_j)$ represents the bounding box of e_j. Recall that $\mathcal{D}(b) = \frac{d(b)}{s(b)}$ is the demand density at boundary b and $d_{prob}(b, e_j)$ is the probability that the Z-edge e_j runs across boundary b. After finding the post node, we generate the probabilistic demands from the two new Z-edges. The process of setting the post node is performed for each backbone wire in every routing tree, which is the first sweep. During this sweep, the backbone wires that have been processed are in Z-edges while those have not been processed are still soft edges. Thus, the demand density $\mathcal{D}(b)$ includes primitive demands from other soft edges and probabilistic demands from other Z-edges.

After post nodes have been selected for all of the backbone wires, all of the demands become probabilistic demands. Based on this improved congestion estimation, we start the second sweep for all backbone wires to specify their routes in the same order as in previous sweep. For each backbone wire, we recompute its post node before fixing the routes of its two Z-edges. A backbone wire appears early in the order list may have a poor post node location in the previous sweep, since this location is chosen according to mostly primitive demands. In this second sweep, this backbone wire has a chance to adjust its post node location from a more accurate congestion information. The procedure of recomputing the post node is the same as in previous sweep. We choose the route for a Z-edge e_j to minimize congestion in term of a cost defined as: $cost(e_j) = \sum_{\forall b \text{ intersecting route of } e_j} \mathcal{D}(b)^2$. The minimum cost route can be found through simple enumeration in a manner similar to calculating the probabilistic demand. After fixing the routes for each backbone wire, its probabilistic demand is replaced by determined demand.

6 Experimental results

Table 1: Description of Test Circuits.

Circuit	# modules	# nets	# pins
ami33	33	85	442
xerox	10	203	696
ami49	49	390	913
test1	-	2100	4576
test2	-	3107	6817

The circuits that we tested includes benchmark suite *ami33*, *ami49* and *xerox* and two sets of randomly generated nets, whose statistics are shown in Table 1.

For comparisons, we also implemented three variations of rip-up-and-reroute (RR) global routing algorithm. In the base version of RR, we initially route each net separately in MVERT but using only solid edges. Then, we rip up every backbone wire in the region with wire overflow, and reroute them through maze routing to minimize congestion cost which is same as for Z-edges except that the demand is determined. Three variations are: RR+B (RR with bends control), RR+T (timing-constrained RR) and RR+B+T (timing-constrained RR with bends control). In order to control the number of bends in RR, we replace the cost in maze routing with a weighted sum of congestion and number of bends. We run the RR+B with several different values of weight and choose the result that provides the least congestion, while assuring that the number of bends for each backbone wire is no greater than five, which is same as our method. In RR+T and RR+B+T, the timing constraints are imposed on the wirelength for each backbone wire in rerouting.

The experimental results are shown in Table 2. There is no timing violations from RR+T, RR+B+T and our method. The maximal number of bends on a backbone wire is five in RR+B, RR+B+T and our method. The rightmost column lists the CPU time in seconds. Since a circuit may include many multi-pin nets, it would be more interesting to evaluate the CPU time for each 2-pin net as a normalized comparison. Column 3 lists the number of backbone wires in each circuits. It is easy to regard these backbone edges as a decomposition into 2-pin nets. Therefore, we can get that the average CPU time for a 2-pin net is 0.2 seconds.

Table 2: Experimental results, *vio* is number of nets with timing violations and *ben* is the maximum number of bends on a backbone wire.

Circuit	Grid size	$\|E\|$	RR+B			RR+T			RR+B+T		Ours		
			$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	*vio*	$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	*ben*	$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	CPU(*s*)
ami33	22 × 36	489	0	1.00	5	1	1.20	12	1	1.20	0	1.00	15.8
xerox	54 × 55	587	0	1.00	6	5	1.14	14	5	1.14	0	0.86	16.1
ami49.1	40 × 41	594	1	1.10	34	0	1.00	18	5	1.10	0	1.00	29.8
ami49.2	52 × 53	593	0	1.00	38	0	1.00	14	51	1.80	2	1.10	25.9
test1	53 × 53	2648	5	1.13	28	0	1.00	17	107	1.12	0	1.00	562
test2	52 × 52	3992	4	1.16	35	0	1.00	20	63	1.14	0	1.00	1013

7 Conclusion

We propose a new approach to improve the quality of global routing. In addition to exploiting timing constrained routing flexibilities, we apply a simple gradual refinement method based on probabilistic congestion estimation, which leads to simultaneous optimization on congestion, timing and number of bends.

References

[1] C. Chiang and M. Sarrafzadeh, "Global routing based on Steiner min-max trees," *IEEE Transactions on CAD*, Vol. 9, No. 12, pp. 1318-1325, December 1990.

[2] B. S. Ting and B. N. Tien, "Routing techniques for gate array," *IEEE Transactions on CAD* , Vol. CAD-2, No. 4, pp. 301-312, October 1983.

[3] J. D. Cho and M. Sarrafzadeh, "Four-bend top-down global routing," *IEEE Transactions on CAD* , Vol. 17, No. 9, pp. 793-802, September 1998.

[4] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," *ISPD*, pp. 19-25, 2000.

[5] J. Huang, X.-L. Hong, C.-K. Cheng and E. S. Kuh, "An efficient timing-driven global routing algorithm," *DAC*, pp. 596-600, 1993.

[6] J. Hu and S. S. Sapatnekar, "A timing-constrained algorithm for simultaneous global routing of multiple nets," *ICCAD*, pp. 99-103, 2000.

[7] J. Hu and S. S. Sapatnekar, "Algorithms for non-Hanan-based optimization for VLSI interconnect under a higher-order AWE model," *IEEE Transactions on CAD*, Vol. 19, No. 4, pp. 446-458, April 2000.

[8] H. Hou, J. Hu and S. S. Sapatnekar, "Non-Hanan routing," *IEEE Transactions on CAD*, Vol. 18, No. 4, pp. 436-444, April 1999.

[9] H.-M. Chen, H. Zhou, F. Y. Yang, H. H. Yang and N. Sherwani, "Integrated floorplanning and interconnect planning," *ICCAD*, pp. 354-357, 1999.