Performance Driven Global Routing Through Gradual Refinement

Jiang Hu
IBM Microelectronics
11400 Burnet Road, MS 4359
Austin, TX 78759, USA
Tel: 512-838-7466
Email: jianghu@us.ibm.com

Sachin S. Sapatnekar *
Department of ECE
University of Minnesota
Minneapolis, MN 55455, USA
Tel: 612-625-0025, Fax: 612-625-4583
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.

Keywords

VLSI, deep-submicron, interconnect, timing-optimization, physical-design, global routing.

*Contact author

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. With interconnects playing an increasingly dominant role in determining circuit performance, it is becoming even more important than ever to perform the global routing task in a methodical, efficient and optimal manner, so that performance-critical nets and long-distance nets can be routed in such a way that constraints on timing, congestion and reliability are met. In this paper, we will propose a new approach to enhance the quality of global routing.

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, i.e., no overflow occurs. Since minimizing congestion is very hard to achieve and is essential for global routing, it has long been a focus of research [1–11] in global routing. In the sequential approach, the nets are routed one after another. In [1], for each net, a minimum weighted Steiner tree spanning the grid graph is sought to minimize the congestion, with the weights being proportional to the density of wires in each routing cell. The sequential approach requires the nets to be routed in some order, on which the quality of the solution depends. As a solution to avoid this ordering problem, the hierarchical method [2–4] recursively splits the routing region into successively smaller parts. At each hierarchical level, all of the nets are routed simultaneously and refined in the next hierarchical level until the lowest level of the hierarchy is reached. Sometimes the entire global routing problem is formulated and solved through linear programming followed by a randomized rounding [5]. Another method is the application of multicommodity flow model [6, 7], in which the fractional solutions are rounded to obtain the routing solutions. For global routing on standard cell designs, the work of [8] proposed an iterative deletion technique to avoid the net ordering problem. The works of [9–11] first route each net independently, then rip up the wires in congested areas and reroute them to spread out the routing density. The rip-up-and-reroute technique is very practical and popular in industrial applications.

When interconnect becomes a performance bottleneck in deep submicron technology, merely minimizing congestion is not enough. In later works [12–14], interconnect delays are explicitly considered

during global routing. In [12], each net is initially routed in SERT-C [15], after which the congested area is ripped up and rerouted by applying a multicommodity flow algorithm locally. In [13], the delay issues are considered more strictly. Beginning with a set of routing trees satisfying timing constraints for each net, a multicommodity flow method is applied to choose a single routing tree for each net, such that the congestion is minimized. At places where overflow occurs, the wires are ripped up and rerouted through a maze routing procedure in which the timing objective is combined with wirelength and congestion. Recently, a timing-constrained global routing scheme is proposed in [14]. This method obtains routing flexibilities under timing constraints and exploits them through a combination of hierarchical bisection and network flow algorithm.

Besides congestion and timing, the number of bends for each wire needs to be limited. A wire bend usually implies a switching of layers, which involves a via resistance that adds to the delay and reduce reliability. In addition, vias will consume more wiring space because their larger pitch requirement. In work of [4], a hierarchical global routing algorithm is proposed to control to number of vias for each wire. There are many other works in routing to minimize the number of vias, especially for MCM and PCB design, such as V4R [16].

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. To the best of our knowledge, this is the first work to optimize all these three objectives simultaneously in global routing.

Similar to the work of [14], 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 [14]. Moreover, we integrate restrictions on wire routes with the refinement so that the number of bends on wires can be bounded. The work of [14] has only limited control over this and there is no obvious extension of the work to bound the number of bends on wires. Imposing the restriction on number of bends naively on the algorithm of [14] would make the bend resource allocation biased to the decisions in higher

hierarchical levels and the routings in lower hierarchical levels may become unnecessarily unachievable. 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.

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 (we omit the net index for simplification without loss of generality) 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 the following terms:

Definition 1: In a routing tree, a *backbone node* is the source node, or a sink node, or a Steiner node with degree greater than 2 (we treat a bend node as a Steiner node with degree two).

Definition 2: In a routing tree $T(V, E)$, a *backbone wire* is a set of consecutively adjoined edges $\{(v, u_1), (u_1, u_2), (u_2, u_3), ...(u_m, w)\}$, where $v, w \in V$ are backbone nodes and $\{u_1, u_2, ...u_m\} \in V$ are not backbone nodes.

If the delay at an arbitrary sink v_a is $t(v_a)$ and the its required arrival time is $RAT(v_a)$, then the delay slack $s(v_a) = RAT(v_a) - t(v_a)$. The timing slack $\mathcal{S}(T^i)$ for a routing tree T^i on the net N^i is the minimum delay slack among all the sinks in this net.

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 $g_i, g_j \in V_G$. 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:

$$f_{ov}(b) = \begin{cases} d(b) - s(b), & d(b) > s(b) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

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.

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 $\mathcal{S}(T^i) \geq 0$, number of bends on each backbone wire is no greater than 5 and the congestion is minimized in terms of $\mathcal{D}_{max}$ and $\mathcal{F}_{ov}$.

3 DELAY ESTIMATION AND ASSUMPTIONS

In this work, we use the π RC model for wires, RC switch model for drivers and Elmore delay model for delay calculation [17, 18] , all of which are common approaches in timing driven routing methods. Although occasional large errors limit the accuracy of the Elmore delay for critical nets [19], it has a role in global routing because of its fidelity [15] and simplicity, and is a reasonable model considering that the routing in global stage is rough and the number of nets may be very large. The extra delay from vias are neglected here because of the same reason, and besides, the number of vias is bounded in our method. The driver resistance, wire capacitance, wire resistance, sink load capacitance and required arrival time (RAT) for each sink are given input to our algorithm. We consider the two-layer routing situation, where one layer is for horizontal wires and the other is for vertical wires. We assume that the per unit length wire capacitance and wire resistance for two layers are very close.

4 ROUTING FLEXIBILITIES UNDER TIMING CONSTRAINTS

In this section, we will explore the routing flexibilities under timing constraints. Usually there are many routing tree topologies that can satisfy the required arrival time (RAT) for each sink in a net, if the RAT are in a reasonable range. For example, both PTree [20] and RAT-Tree [21] can generate a set of routing topologies satisfying timing constraints. Such timing-constrained routing flexibilities are exploited in the global routing work in [13]. In this work, we will use some other types of routing

flexibilities including: soft edges, Z-edges, slideable Steiner nodes and edge elongation. Most of these are proposed in [19, 14] and will be summarized here for completeness.

4.1 Soft Edges

Since a routing tree will be built in rectilinear space finally, each edge must be either horizontal or vertical. Thus two nodes $v_i, v_j \in V$ with their coordinates $x_i \neq x_j$ and $y_i \neq y_j$ are usually connected with one horizontal edge and one vertical edge in an L-shape. Between any two such nodes, there are always two L-shaped connections and many other connections with same wirelength if the number of bends is not restricted. They can be regarded to have same delay performance in global routing, since the routes are specified coarsely, especially when the number of bends is limited. This timing-constrained routing flexibility is expressed through the concept of a soft edge defined as follows.

Definition 3: A *soft edge* is an edge connecting two nodes $v_i, v_j \in V$, such that:

1. $x_i \neq x_j$ and $y_i \neq y_j$,
2. its edge length $l_{ij} = |x_i - x_j| + |y_i - y_j|$,
3. the precise edge route between v_i and v_j is not determined.

We will refer to the traditional edges in a rectilinear tree with fixed orientations as *solid edges*. It has been shown in [19, 14] that the concept of a soft edge is useful in both single net performance-driven and multi-net global routing.

4.2 Z-edges

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.

4.3 Slideable Steiner Node(SSN)

The work of MVERT [22] shows that using non-Hanan Steiner nodes can help to improve the routing tree performance, if the objective is to minimize total wirelength subject to timing constraints. In [19],

this non-Hanan optimization technique is extended to routing in soft edges.

We will use the example in Fig. 1, to describe another timing-constrained routing flexibility. If the optimal Steiner node v' among nodes v_i, v_j and v_k is off the Hanan grid as depicted in Fig. 1, the location of v' is specified by the Manhattan distance from the upsteam end v_i to v' in [22, 19]. In fact, there are often many points with the same distance. The set of locations for a distance form a segment of locus as illustrated by the thickened segment in Fig. 1. When we slide the Steiner node v' along this locus, the length of its incident edges are preserved and so does the delay at each sink. Similar to the rationale for soft edges, we only specify this locus instead of a point for this Steiner node and call it as *slideable Steiner node* (SSN).

4.4 Edge Elongation

Besides soft edge, Z-edges and slideable Steiner nodes (SSN), we use another simple timing-constrained routing flexibility. After performance-driven routing for a net, it is possible that the timing slack of the routing tree is still positive. This positive slack can be consumed through edge elongation to provide more routing flexibilities under timing constraints.

The maximum allowed elongation Δl_{ij} for routing edge e_{ij} can be computed under the Elmore delay model. The length of a routing path from source v_0 to a node $v_i \in V$ is denoted as p_i, and the shared path length for two nodes $v_i, v_j \in V$ from the source is expressed as p_{ij}. For any sink $v_k \in V$, we can compute the maximum Δl_{ij} such that the delay slack $s(v_k)$ is non-negative. If $v_k \notin T_i$, i.e., v_k is not in downstream of v_i,

$$\Delta l_{ij} = \frac{s(v_k)}{(R_d + \hat{r}p_{ik})\hat{c}} \tag{2}$$

where R_d is the driver resistance, $\hat{r}$ and $\hat{c}$ are the wire resistance and capacitance per unit length. If $v_k \in T_i$, Δl_{ij} satisfies the following equation:

$$s(v_k) = f(\Delta l_{ij}) = (R_d + \hat{r}p_i)\hat{c}(\Delta l_{ij}) + \frac{1}{2}\hat{r}\hat{c}(2l_{ij}(\Delta l_{ij}) + (\Delta l_{ij})^2) + \hat{r}(\Delta l_{ij})C_j, \tag{3}$$

where C_j is the total downstream capacitance from v_j. This equation can be solved to obtain the Δl_{ij}. In the case of double roots for this equation, we choose the one where the slope of function $f(\Delta l_{ij})$ is positive, since the delay slack should be monotonically increasing with respect to the allowed elongation.

We compute Δl_{ij} for all the sinks in the routing tree and choose the minimum value as a safe value. Note that different edge may have different value of maximum allowed elongations.

5 APPROXIMATED CONGESTION ESTIMATION

In addition to the traditional congestion metrics, we use a couple of other approximate estimation methods during different phases of global routing, all of which will be introduced in details as follows.

5.1 Primitive Demands

The *demand* defined in section 2 is based on the the wire routes in solid edges. However, we arrive at a rough estimation of the congestion from soft edges. We use the concept of *primitive demand* to indicate the possibility of wires crossing a grid edge. This is demonstrated in the example in Fig. 2, in which the dashed segments represent the tessellation by the grid graph, G. The bounding box for a soft edge e_{ij} is obtained as the dotted rectangles and denoted as B_{ij}. In Fig. 2(a), there are three vertical boundary segments at $x = x_2$ overlapping with B_{ij} and only one of them is crossed by e_{ij} in the final route. Therefore, we define the primitive demand incurred by e_{ij} over the grid edge corresponding to each of these three boundary segments as $\frac{1}{3}$. Similarly, each grid edge for a thickened horizontal (vertical) boundary segment in Fig. 2(a) has a primitive demand of $\frac{1}{4}(\frac{1}{3})$. We refer to the demand, as defined in section 2, as the *determined demand*, which actually is a special case of the primitive demand. A primitive demand for a soft egde is equivalent to its determined demand when its value aggregates to 1, as in Fig. 2(b). If we denote the primitive demand incurred by routing edge e_{ij} over grid edge b as $d_{prim}(b, e_{ij})$, then:

$$d_{prim}(b) = \sum_{\forall B_{ij} \text{ intersecting } b} d_{prim}(b, e_{ij}) \tag{4}$$

Definition 4: If the bounding box B_{ij} of an edge e_{ij} passes through m rows and n columns in the grid graph G, the *primitive demand* $d_{prim}(b, e_{ij})$ from e_{ij} on each grid edge b corresponding to a vertical(horizontal) boundary overlap with B_{ij} is $\frac{1}{m}(\frac{1}{n})$.

5.2 Probabilistic Demands

The possible routes of a Z-edge are easier to enumerate and a congestion estimation better than the *primitive demand* can be obtained accordingly. In this scenario, we adopt a probabilistic estimation which is similar to [23, 24] and will be illustrated in the example in Fig. 3. Without loss of generality, we can arbitrarily specify one end of the soft edge as source node v_s and the other end as target node v_t, and denote the grid cell in which they are located as (r_s, c_s) and (r_t, c_t), respectively. We use r and c to represent row and column indices. Through simple enumeration, we can conclude that:

Lemma 1: *There are* $Z = |r_t - r_s| + |c_t - c_s|$ *monotone routes with number of bends no greater than* 2 *between two grid cell* (r_s, c_s) *and* (r_t, c_t) *in a grid graph.*

For example, there are seven possible route for the Z-edge in Fig. 3. In our congestion estimation, 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 the soft edge, again through simple enumeration. To simplify the description, we initially consider only those routes that leave source node v_s horizontally toward the target node. Obviously, there are $|c_t - c_s|$ such routes which are depicted in Fig. 3. For each horizontal grid edge above the grid cells $\{(r_i, c_i) | r_s \leq r_i < r_t, c_s < c_i \leq c_t\}$, there is one route across it. Thus, in the example of Fig. 3, the probability that the soft edge runs across the horizontal grid edge above cell (r_1, c_2) is $1/7$. For each vertical grid edge to the right of grid cell (r_s, c_j), $c_s \leq c_j < c_t$, the probability is $|c_t - c_j|/Z$. For example, the probability that the wire run across the vertical grid edge to right of grid cell (r_0, c_1) is $3/7$. The probability at other grid edge can be counted similarly. Based on these probabilities, we define the probabilistic demand as follows:

Definition 5: The *probabilistic demand* from a Z-edge e_{ij} to a grid edge b is the probability that Z-edge run across this grid edge, and is denoted as $d_{prob}(b, e_{ij})$.

6 ALGORITHMS

6.1 Algorithm Motivation

It is well-known that even minimizing only congestion for only 2-pin nets is an NP-complete problem, and considering timing constraints and the number of bends makes the problem even harder. 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. The strategy here is to refine the route gradually according to available congestion information, even this information is ambiguous.

We can compare the underlying mechanism with the sequential and rip-up-and-reroute approach. In the sequential approach, the earlier routing steps are performed without any knowledge of the locations of the subsequent nets, and are therefore somewhat blindfolded. The routing of nets that are considered later is based on the routes of previous nets, which may be suboptimally placed, due to this false feedback. Similarly, rip-up-and-reroute, which proceeds according to the locations of other nets, may also be suboptimal. Due to its iterative nature, rip-up-and-reroute method has the ability to correct the false feedback gradually and may become successful in reducing congestion after many iterations. However, the efficacy of rip-up-and-reroute may be hindered if constraints on timing and the number of bends are imposed.

Fortunately, our method can obtain some rough congestion estimation based on the locations of soft edges after phase one. This estimation is by no means completely accurate, but it is better than no feedback or incorrect feedback. Since we know that this estimation is not entirely accurate, we will not fix the route completely in one step. Instead, we will settle a part of the route to obtain a better congestion estimation and complete the routes gradually to avoid blindfolded or incorrect decision.

6.2 Algorithm

Our algorithm includes two phases:
(I) Timing-driven routing for each net individually without considering congestion or number of bends.
(II) Specifying the route for each backbone wire obtained in phase I so that congestion is minimized subject to timing constraints and bends constraints.

In phase I, we route each net through the MVERT [22] 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.

After phase I, we can obtain a rough estimation of congestions through the concept of *primitive demand* defined in section 5.1. The first step in phase II is to fix the position of each SSN (Slideable Steiner Node) to minimize the peak primitive demand density. Recall that 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. An example of a sliding Steiner node is illustrated in Fig. 4, and it can be seen that the bounding boxes of the incident edges, denoted by B_i, B_j and B_k, are changed after the sliding of the SSN. In Fig. 4, when the Steiner node v' is slid toward northwest from (a) to (b), the bounding box B_i becomes thinner and taller and the vertical (horizontal) primitive demand incurred by (v_i, v') on each horizontal (vertical) cell boundary overlaps with B_i may be larger (smaller). The effect of this move on B_j and B_k is the opposite. 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 the backbone wire has 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. For example, in Fig. 5(a), a post node represented by a small circle is inserted into the backbone wire represented by a dotted curve. Before choosing the grid cell for the post node, we need to choose the candidate grid cells that will be considered. The routing flexibility from edge elongation is utilized here. For a backbone wire e_{ij} with edge length l_{ij}, we calculate the maximum allowed elongation Δl_{ij} under timing constraints as in section 4.4. If we insert a post node v_k into e_{ij}, the location of v_k must satisfy $l_{ik} + l_{kj} \leq l_{ij} + \Delta l_{ij}$. In Fig. 5(a), the candidate grid cells are shaded.

Similarly, a formerly solid edge can also be elongated to reduce congestion. Thus, we see that a solid edge is not exactly solid any more, and can have some routing flexibility as well. Note that the maximum allowed elongation for each backbone wire is calculated dynamically, since the allowed elongations for backbone wires in a same routing tree depend on each other.

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. In our implementations, we set this bound to five. 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 restrictions on timing and number of bends on wires. Moreover, we can obtain a better estimation of the congestion by using the probabilistic demand defined in section 5.2. We choose the post node so as to minimize the congestion cost for the two sub-edges. Before setting the post node for a backbone wire e_i, we need to remove the primitive demand generated from the soft edge of e_i. Then, the congestion cost of a subedge e_j is defined as:

$$cost(e_j) = \sum_{\forall b \text{ intersecting bounding box of } e_j} \mathcal{D}(b)^2 \times d_{prob}(b, e_j). \quad (5)$$

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. In the example in Fig. 5(b), the cost of edge e_1 is the summation of the estimated costs from thickened grid edges. Setting a post node will not exclude the case that a backbone wire is in a direct L-shaped connection, since it is possible that the post node is the bend node and two Z-edges are straight solid edges.

After setting 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 Z-edges while those have not been processed are still soft edges. Thus, in the congestion cost computation, both primitive and probabilistic demand may co-exist at a same time. Therefore, we multiply the primitive demand by a penalty coefficient of less than one to make its contribution weaker than that of the probabilistic demand.

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:

$$cost(e_j) = \sum_{\forall b \text{ intersecting route of } e_j} \mathcal{D}(b)^2. \quad (6)$$

The minimum cost route can be found through simple enumeration in a manner similar to calculating the probabilistic demand in section 5.2. After fixing the routes for each backbone wire, its probabilistic demand is replaced by determined demand. The complete algorithm is called *ZigPuzzle* and summarized in Fig. 6.

7 EXPERIMENTAL RESULTS

We have implemented ZigPuzzle global routing in C++ and conducted experiments on a SUN Ultra-10 workstation. The circuits that we tested includes benchmark suite *ami*33, *ami*49 and *xerox* and three sets of randomly generated nets, whose statistics are shown in Table 1. For some circuits, more than one set of netlists is obtained through different placement. For example, there are two sets of netlists for circuit of *ami49* and *xerox*. Each result is measured in terms of two congestion metrics: total overflow $\mathcal{F}_{ov}$ and maximum demand density $\mathcal{D}_{max}$.

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 same as in phase I of ZigPuzzle but using 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 in equation 6 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 best congestion, while assuring that the number of bends for each backbone wire is no greater than five, which is same as in ZigPuzzle. In RR+T, the timing constraints are imposed on the wirelength for each backbone wire. Both of these methods are combined in obtaining

the results for the RR+B+T case.

The experimental results are shown in Table 2. Column 4 to 6 are results from RR+B, whose number of bends is bounded to be no more than five for each backbone wire. Its congestion results are generally good with limited wire overflows in large circuits. However, it does not have the capability to satisfy timing constraints and there are always a number of nets have timing violations as indicated in column 6. The results from RR+T is given in column 7 to 9, where the congestion is good subject to timing constraints, but the maximal number of bends per backbone wire can be very large (in a range of $12 - 20$). When we impose both timing and bends constraints directly on RR, the results in column 10 and 11 shows that the congestion can be very poor especially in large circuits. On the other hand, our ZigPuzzle can optimize congestion subject to timing and bend constraints, and works well in both small and large circuits. In the last row of the table, the results from RR are compared with that of ZigPuzzle.

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 the average CPU time for a 2-pin net is 0.2 seconds.

In our approach, we use primitive demand to estimate the congestions from soft edges. Instead of this, we can use a probabilistic estimation similar to that in section 5.2 for soft edges by using the technique described in [25], and it may initially seem that this would provide a more accurate estimate of congestion. We have also performed experiments using this approach. However, we find that this approach takes a longer CPU time and often produces worse results. This increased CPU is not unexpected, since the computation related to probabilistic estimation will be more computationally intensive than primitive demand calculations, especially when the allowed number of bends is large. In addition, a backbone wire could be elongated at a later stage, and therefore, the increased accuracy obtained from probabilistic estimation at current stage may become meaningless later. The spirit of primitive demand is to catch a rough estimation in a quick operation and our experiments have demonstrated that it is better than a more sophisticated measure at the early stage of global routing.

8 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 Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 12, pp. 1318-1325, December, 1990.

[2] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No. 4, pp. 223-234, October, 1983.

[3] M. Marek-Sadowska, "Global router for gate array," *Proceedings of the IEEE International Conference on Computer Design*, pp. 332-337, 1984.

[4] J. D. Cho and M. Sarrafzadeh, "Four-bend top-down global routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 9, pp. 793-802, Sep. 1998.

[5] P. Raghavan and C. D. Thompson, "Multiterminal global routing: a deterministic approximation scheme," *Algorithmica*, Vol. 6, pp. 73-82, 1991.

[6] R. C. Carden IV, J. Li and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 2, pp. 208-216, February, 1996.

[7] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," *Proceedings of the ACM International Symposium on Physical Design*, pp. 19-25, 2000.

[8] J. Cong and B. Preas, "A new algorithm for standard cell global routing," *Integration-The VLSI Journal*, Vol. 14, pp. 49-65, 1992.

[9] B. S. Ting and B. N. Tien, "Routing techniques for gate array," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No. 4, pp. 301-312, October, 1983.

[10] R. Nair, "A simple yet effective technique for global wiring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-6, No. 2, pp. 165-172, October, 1987.

[11] K. W. Lee and C. Sechen, "A global router for sea-of-gate circuits," *Proceedings of the European Design Automation Conference*, pp. 242-247, 1991.

[12] D. Wang and E. S. Kuh, "Performance-driven interconnect global routing," *Proceedings of the IEEE Great Lakes Symposium on VLSI*, pp. 132-136, 1996.

[13] J. Huang, X.-L. Hong, C.-K. Cheng and E. S. Kuh, "An efficient timing-driven global routing algorithm," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 596-600, 1993.

[14] J. Hu and S. S. Sapatnekar, "A timing-constrained algorithm for simultaneous global routing of multiple nets," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 99-103, 2000.

[15] K. D. Boese, A. B. Kahng, B. A. McCoy and G. Robins, "Near-optimal critical sink routing tree constructions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 12, pp. 1417-36, Dec. 1995.

[16] K.-Y. Khoo and J. Cong, "An efficient multilayer MCM router based on four-via routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 10, pp. 1277-1290, October 1995.

[17] W. C. Elmore, "The transient response of damped linear network with particular regard to wide-band amplifiers," *Journal of Applied Physics*, Vol. 19, pp. 55-63, 1948.

[18] J. Rubinstein, P. Penfield and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-2, No. 3, pp. 202-211, July 1983.

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

[20] J. Lillis, C. K. Cheng, T. T. Lin and C. Y. Ho, "New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 395-400, Jun. 1996.

[21] J. Cong and C. K. Koh, "Interconnect layout optimization under higher-order RLC model," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 713-720, 1997.

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

[23] H.-M. Chen, H. Zhou, F. Y. Yang, H. H. Yang and N. Sherwani, "Integrated floorplanning and interconnect planning," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 354-357, 1999.

[24] P. Sarkar, V. Sundararaman, C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," *Proceedings of the ACM International Symposium on Physical Design*, pp. 186-191, 2000.

[25] Kusnadi and J. D. Carothers, "A method of measuring nets routability for MCM's general area routing problems," *Proceedings of the ACM International Symposium on Physical Design*, pp. 186-192, 1999.

JIANG HU received his B.S. in optical engineering from Zhejiang University, China in 1990, M.S. in physics from University of Minnesota in 1997, and Ph.D. in electrical engineering from University of Minnesota in 2001. Currently, he is a CAD engineer at IBM Microelectronics Division. His research interest is on VLSI physical design automation, especially interconnect routing and optimization.

SACHIN SAPATNEKAR received his B.Tech. from Indian Institute of Technology, Bombay in 1987, M.S. from Syracuse University in 1989, and Ph.D. from University of Illinois at Urbana-Champaign in 1992. From 1992 to 1997, he was an assistant professor in Department of ECE at Iowa State University. He is currently an associate professor in Department of ECE at University of Minnesota. His research interest is on VLSI CAD, concentrating on timing, power and layout issues. He has served as an Associate Editor for the IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. He is a recipient of the NSF Career Award and Best Paper awards at DAC97 and ICCD98.

Figure caption list

Figure 1: An example of a slideable Steiner node(SSN).

Figure 2: 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.

Figure 3: Enumerate routes with number of bends less than 3 to obtain probabilistic demand.

Figure 4: When an SSN slides along its locus, the bounding boxes of its incident edges change as well as the primitive demands.

Figure 5: Examples for setting post node for a backbone wire.

Figure 6: Algorithm.

Table caption list

Table 1: Description of Test Circuits.

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

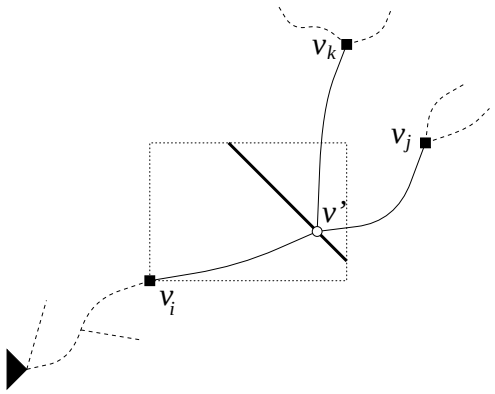

Figure 1:

Jiang Hu

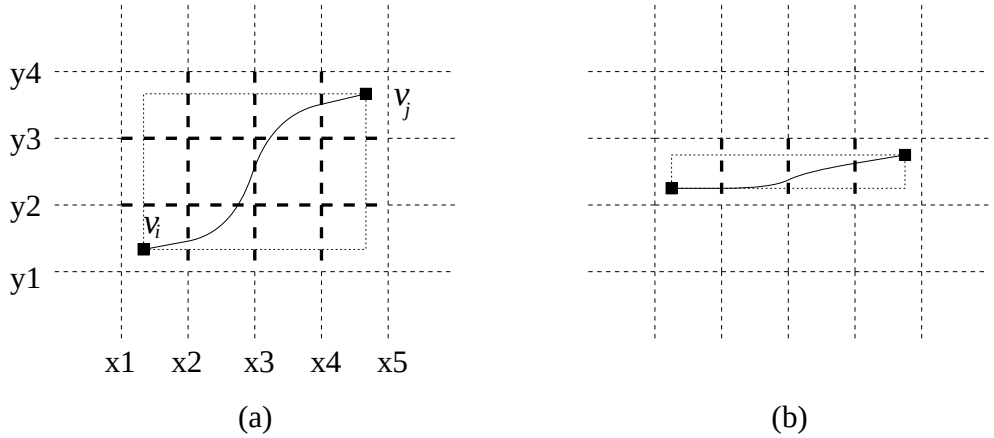

Figure 2:

Jiang Hu

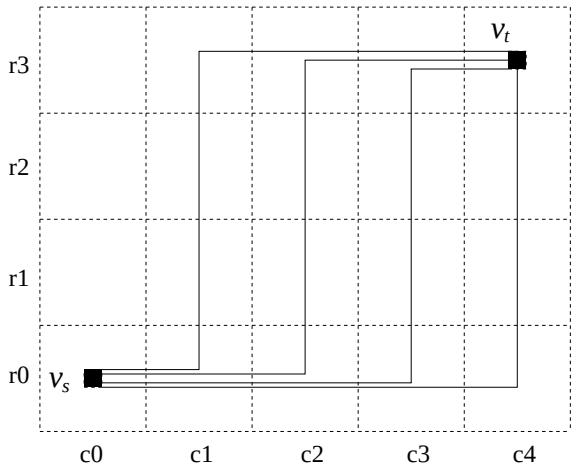

Figure 3:

Jiang Hu

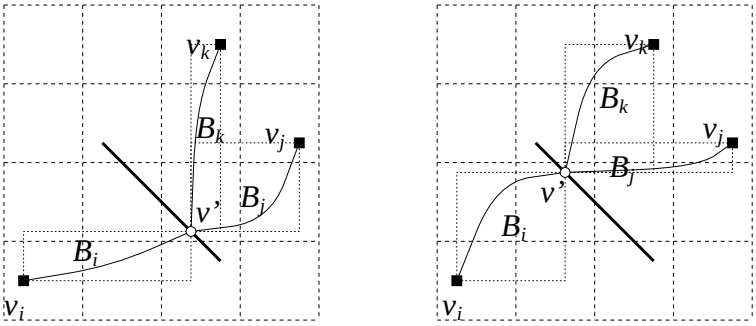

Figure 4:

Jiang Hu

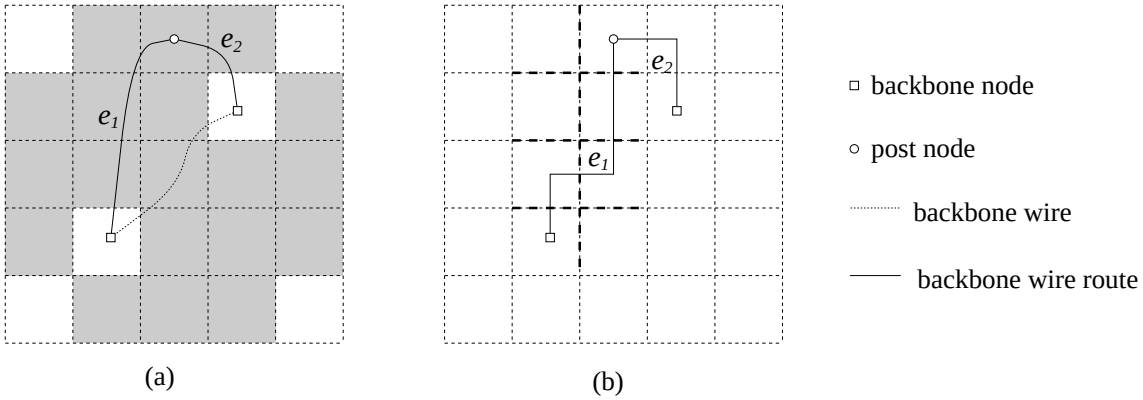

Figure 5:

Jiang Hu

Algorithm: ZigPuzzle($\mathcal{N}, G$)
Input: A set of nets $\mathcal{N}$, grid graph $G(V_G, E_G)$
Output: T^i for each N^i min congestion
s.t. timing constraints, ≤ 5 bends per backbone wire
Phase I
1. For each $N^i \in \mathcal{N}$
2. $T^i \leftarrow$ MVERT(N^i)
Phase II
1. Generate primitive demand for each backbone wire
2. Fix all of the SSNs, min peak demand density
3. For each tree T^i
4. For each backbone wire $e^i_{jk} \in T^i$
5. Remove its primitive demand
5. Compute max elongation Δl_{jk} without timing violation
6. Find candidate grid cells for post node
7. Search a post node, min congestion cost
8. Generate probabilistic demand from two Z-edges
9. For each tree T^i
10. For each backbone wire $e^i_{jk} \in T^i$
11. Recompute its post node
12. Remove its probabilistic demand
13. Fix routes for two Z-edges, min congestion cost
14. Generate determined demands from two Z-edges

Figure 6:

Table 1:

Circuit	# modules	# nets	# pins
$ami33$	33	85	442
$xerox$	10	203	696
$ami49$	49	390	913
$test1$	-	1109	2464
$test2$	-	2100	4576
$test3$	-	3107	6817

Jiang Hu

Table 2:

			RR+B			RR+T			RR+B+T		ZigPuzzle		
Circuit	Grid size	$\|E\|$	$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	vio	$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	ben	$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	$\mathcal{F}_{ov}$	$\mathcal{D}_{max}$	CPU(s)
ami33.1	22 × 36	489	0	1.00	5	1	1.20	12	1	1.20	0	1.00	15.8
xerox.1	54 × 55	587	0	1.00	6	5	1.14	14	5	1.14	0	0.86	16.1
xerox.2	54 × 61	571	0	1.00	6	2	1.06	12	2	1.06	2	1.06	23.3
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	52 × 52	1468	4	1.29	34	0	1.00	17	58	1.21	1	1.07	207
test2	53 × 53	2648	5	1.13	28	0	1.00	17	107	1.12	0	1.00	562
test3	52 × 52	3992	4	1.16	35	0	1.00	20	63	1.14	0	1.00	1013
Average	47 × 50	1368	1.8	1.09	23	1	1.05	16	36.5	1.22	0.6	1.01	237
Comp			3×	1.05×		1.67×	1.03×	3.2×	61×	1.20×			

Jiang Hu