Hindawi Publishing Corporation
VLSI Design
Volume 2009, Article ID 537341, 13 pages
doi:10.1155/2009/537341

Research Article

A Multilevel Congestion-Based Global Router

Logan Rakai,[1] Laleh Behjat,[1] Shawki Areibi,[2] and Tamas Terlaky[3]

[1] *Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Calgary, AB, Canada T2N 1N4*
[2] *School of Engineering, University of Guelph, Guelph, ON, Canada N1G 2W1*
[3] *Department of Industrial and Systems Engineering, P. C. Rossin College of Engineering and Applied Science, Lehigh University, Bethlehem, PA 18015, USA*

Correspondence should be addressed to Logan Rakai, lmrakai@ucalgary.ca

Received 17 February 2009; Revised 13 July 2009; Accepted 2 September 2009

Recommended by David S. Kung

Routing in nanometer nodes creates an elevated level of importance for low-congestion routing. At the same time, advances in mathematical programming have increased the power to solve complex problems, such as the routing problem. Hence, new routing methods need to be developed that can combine advanced mathematical programming and modeling techniques to provide low-congestion solutions. In this paper, a hierarchical mathematical programming-based global routing technique that considers congestion is proposed. The main contributions presented in this paper include (i) implementation of congestion estimation based on actual routing solutions versus purely probabilistic techniques, (ii) development of a congestion-based hierarchy for solving the global routing problem, and (iii) generation of a robust framework for solving the routing problem using mathematical programming techniques. Experimental results illustrate that the proposed global router is capable of reducing congestion and overflow by as much as 36% compared to the state-of-the-art mathematical programming models.

1. Introduction

VLSI physical design is a way of producing a physical layout for a circuit from an abstract set of circuit components, connections, and requirements. Global routing is an important phase of VLSI physical design which determines the approximate pathways of wires, or *interconnects*, in the layout. The importance of this phase has increased since the interconnect delay exceeds device delay at modern nanometer and gigahertz process nodes [1]. Interconnect is responsible for the majority of dynamic power consumption and is projected to become responsible for 80% of the total power dissipation by 2012 if no changes in design philosophy are made [2]. These realities brought about the 2007 and 2008 International Symposium of Physical Design (ISPD) Contests on global routing [3, 4] which provided the impetus for research innovation in the area.

As circuit design complexity has increased, developing global routing solutions that eliminate overflow and reduce congestion has become a key goal of global routers. Overflow occurs when the number of wires in a channel is higher than the maximum number of wires that can be physically put in the channel. When routing channels are overflowing, the layout cannot be fabricated. Congestion happens when the number of wires in a routing area is very close to the maximum. High congestion can deteriorate signal integrity and increase delay uncertainty [5]. A considerable amount of work has been done on avoiding congestion in global routing, such as [6, 7], and several methods exist for predicting congestion before global routing [8–10].

In this paper, a mathematical programming-based global router is proposed. The router is aimed at minimizing the number of unrouted nets, or equivalently, maximizing the number of routed nets without creating overflow. The problem has been formulated as an Integer Linear Programming (ILP) problem with congestion-based weight coefficients augmented in the objective function to preferentially alleviate congestion. An accurate congestion map generation technique is proposed that can quickly model congestion. The congestion map generation technique is ideally suited for ILP-based routers and can be adapted to any objective. It is proposed to use the congestion map to produce a congestion-based hierarchy to perform the routing. The method for producing a hierarchy could be used to aid any concurrent

routing method in reducing congestion. This method is able to alleviate the most urgent congestion problems over the entire circuit and is not limited to solving local instances.

The remainder of the paper is organized as follows. The relevant global routing background and the motivations are given in Section 2. In Section 3, main ideas of the proposed router are described. Experimental results validating the performance of the proposed router and comparisons between the proposed and other modern academic routers are shown in Section 4. The paper is concluded with a summary of the proposed router and possible future improvements in Section 5.

2. Global Routing Background

2.1. Global Routing Methodologies. Global routing can be performed by routing nets sequentially or concurrently. In sequential routing nets are routed one-at-a-time without explicit consideration of nets to be routed in the future. Focusing on a single net is conceptually easier but problems can arise based on the order in which the nets are routed. For example, certain edges can become over congested at the beginning of the routing process and longer routes need to be made to avoid these edges, which in turn can result in excessive length. Because of the so-called *net ordering problem*, sequential techniques cannot provide information on whether a feasible solution exists nor predict the solution quality. A few examples of sequential global routers that have taken part in ISPD global routing contests include FGR [11], NTHU-Route [12], NTUgr [13], and FastRoute [14].

In concurrent routing, mathematical programming is used to route all nets simultaneously. This is accomplished by producing a set of paths or trees for each net prior to solving the problem. The mathematical program selects the trees that optimize a given objective while obeying the routing constraints; so ordering problems are avoided. Because of the advances in mathematical programming solvers, several concurrent global routers have been proposed recently [8, 15, 16]. BoxRouter [8] is an ILP-based router that uses progressive box expansion to formulate a sequence of ILPs that incrementally route the circuit. BoxRouter was competitive in both the 2007 and 2008 ISPD routing contests. Sidewinder [15] is a flat router that moderates the ILP size by only selecting two candidate trees from a potentially very large group of initial candidates. Multiobjective Router, presented in [16], has a variety of routing objectives to choose from including routing the maximum number of two-pin subnets without creating overflow.

2.2. Global Routing Steps. Regardless of the methodology, concurrent or sequential, almost all global routers use the following steps: (1) graph representation, (2) tree construction, and (3) congestion alleviation. These important steps are briefly described below.

2.2.1. Graph Representation. The input to the global routing is the placement solution in which the location of the components on the layout surface is specified. The first step in global routing is to overlay a grid on this placement solution in which each grid cell becomes a vertex and each boundary between grid cells becomes an edge. The number of routing tracks that a grid cell encompasses determines the capacity of an edge. In modern designs, this capacity can be further reduced because of the presence of blockages, such as IP blocks. To provide a global routing solution for 3-dimensional designs, the 3-dimensional grid is commonly collapsed into a 2-dimensional grid and a layer assignment process expands the solution back to three dimensions. This process is illustrated in Figure 1. A variety of layer assignment methods exist in literature [8, 11, 17]. In this paper we are only concerned with the 2-dimensional solution of the problem.

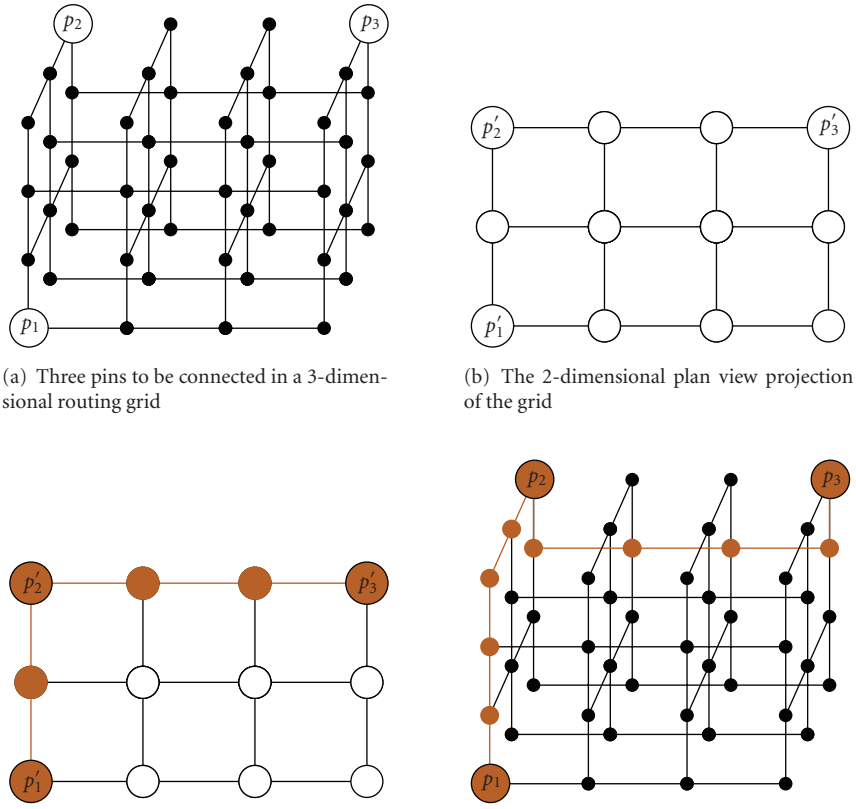
2.2.2. Tree Construction. At the core of any global routing is the tree construction algorithm that produces one or more paths for each net. Trees are used to represent a route for a net. Rectilinear minimum spanning trees (RMSTs) are frequently used to produce reasonable trees at low runtime cost. RMSTs can be constructed with a standard algorithm, such as Prim's [18]. However, the length of trees can be larger than the minimum length possible. Rectilinear Steiner minimal trees (RSMTs) will produce minimum length trees [19]. RSMT construction is NP-hard [20] but high-quality approximate algorithms such as FLUTE [21] exist that produce near optimal RSMTs quickly.

Depending on the number of pins a net contains, different strategies can be used to produce trees for it. If a net has two pins, then the lengths of a rectilinear minimum spanning tree and a rectilinear Steiner minimal tree are the same. In addition, many different paths can be easily made for such trees. However, when a net contains many pins, tree generation can become very time consuming, and usually the global router can afford to only produce a small number of trees for these nets. In order to avoid large runtimes associated with producing several trees for multipin nets, in this paper, the technique used in [16] is employed where nets with more than two pins are decomposed to several two-pin connections. These two pin connections are considered as individual nets during the routing.

2.2.3. Congestion Alleviation. Initial global routing solutions often contain overflow, making the routing illegal. Several strategies exist for congestion and overflow alleviation. Some of these strategies are described in the following.

Edge Shifting [22]. Once trees are constructed for all or most of the nets, congestion in edges can be alleviated by shifting segments of a tree out of the area of congestion. Wirelength may increase as a result of this.

History-Based Costs [23]. Edge costs during an iteration of refinement are based not only on current demand for an edge but also on all the historical demand. By inflating the cost of an edge according to its historical precedent, problematic edges are avoided.



(a) Three pins to be connected in a 3-dimensional routing grid

(b) The 2-dimensional plan view projection of the grid

(c) The 2-dimensional route

(d) The route after layer assignment

Figure 1: An example of converting a placement into a routing grid graph [17].

Rip-up and Reroute [24]. A framework for iteratively improving solutions, known as rip-up and reroute (RRR), can be used to iteratively improve the solutions by removing routes that pass through congested edges and finding new routes for them using less utilized edges.

Congestion Maps [25]. An often employed strategy to aid a global router in avoiding the creation of congestion is the production of a congestion map. Congestion maps can allow the router to avoid overflow early on in the routing and converge to a legal solution more quickly. For concurrent routing, this can reduce the number of trees that need to be generated, hence reducing the time to solve the problem. A map should be obtained quickly to ensure a decrease in overall runtime. This implies a trade-off in the accuracy of the congestion prediction and time spent generating it.

2.3. ILP-Based Formulation. In ILP-based global routing, for a net $n_i \in N$, where N is the set of all nets to be routed, a set of corresponding trees T_i is produced. An individual tree $t_{i,j}$ is referred to with a dual index. The first index, $1 \le i \le |N|$, refers to the net that the tree produced for, while the second index, $1 \le j \le |T_i|$, indicates the number of the tree in the set produced for the net n_i. To generate an ILP formulation, each tree $t_{i,j}$ is assigned a binary variable $x_{i,j}$. The binary variable $x_{i,j}$ is one if the tree is selected in the final solution and zero otherwise. The general format of an ILP problem is

$$\begin{aligned} &\text{minimize} \quad f(\cdot) \\ &\text{subject to} \quad \sum_{t_{i,j} \in T_i} x_{i,j} = 1, \quad \forall n_i \in N, \\ &\qquad \sum_{\{i,j \mid e_k \in t_{i,j}\}} x_{i,j} \le c_{e_k}, \quad \forall e_k \in E, \\ &\qquad x_{i,j} \in \{0,1\}, \quad \forall n_i \in N, \ \forall t_{i,j} \in T_i, \end{aligned} \tag{1}$$

where $f(\cdot)$ is cost function of the optimization problem, c_{e_k} is the capacity of edge $e_k \in E$, and E is the set of all routing grid edges. The first set of constraints in (1) are referred to as the *exclusivity constraints* and ensure that only one tree is selected for each net. The second set of constraints are the *edge capacity constraints* and ensure that the demand for each edge remains lower or equal to the maximum capacity. Finally, the last set of constraints, the *integrality constraints*, restrict the variables being binary.

Several objective functions have been proposed for the global routing problem such as $f(\mathbf{x}) = \mathbf{l}'\mathbf{x}$, where $\mathbf{l}$ is a vector with each entry set to the length of the corresponding tree [26]. In [27], minimization of the maximum edge demand is presented as an objective function. This model replaces all of the edge capacities with a single variable

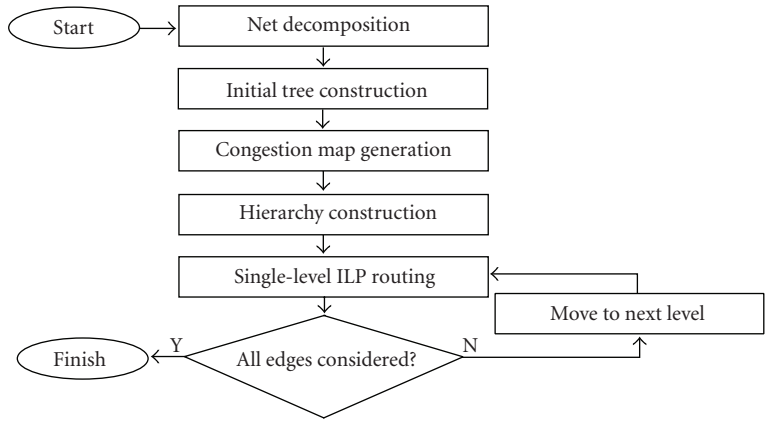


Figure 2: The flow chart specifying the operation of the proposed router.

and the objective is to minimize that capacity variable. A weighted combination of objectives including minimization of wirelength, congestion, and number of vias (number of bends in the routes) is proposed in [28]. In [8, 15, 16], the objective is changed to maximize the number of routed nets. For all of the above objectives except for the minimization of the maximum edge demand, the objective function can be written as a linear combination of the binary variables: $f(\cdot) = \mathbf{w}'\mathbf{x}$, where $\mathbf{w}$ is the vector of the costs associated with the trees. This cost might include wirelength and via count amongst other measures of tree quality.

2.4. Hierarchical Routing. A concurrent method can become sequential by imposing a hierarchy amongst the nets [29]. The number of levels in the hierarchy loosely determines the number of nets to be routed at each level and, thus, the number of variables in each of the programs. Having a small number of nets to route allows a large number of trees to be considered. BoxRouter [8] uses a hierarchical approach that routes nets in a box encompassing the most congested region in a design. The box is iteratively expanded until it contains the entire routing grid. Another paradigm in multilevel routing is to successively coarsen the routing grid and to route nets on the coarser grids first. These routes are refined as the grid is uncoarsened back to the initial, fine grid graph. This strategy is used in [30], which has an advanced resource reservation method to ensure fine nets can be routed.

2.5. The Proposed Router's Motivation. Since concurrent routers perform routing without ordering nets, they should have an advantage over sequential routing in dealing with congestion. However, existing concurrent methods have not sufficiently addressed the problem of alleviating congestion. Sidewinder [15] uses a dynamically updated congestion map to avoid congested areas in producing trees at each iteration. This is a time-consuming strategy that contributes to the prohibitively large runtimes of the router. In the Multiobjective Router [16] congestion is not considered during the formulation of the ILP problem. BoxRouter's [8] box expansion allows a predictably congested area to be routed before expanding the box. This is insufficient for removing congestion and sequential RRR is performed after each expansion.

In the proposed router a new multilevel strategy is developed that uses congestion maps generated by solving a relaxed ILP formulation to define the hierarchy. This strategy allows the router to better remove congestion while being still within a concurrent framework and removes reliance on heuristic sequential techniques.

3. The Proposed ILP-Based Router to Effectively Maximize Routed Nets

In this section, an ILP-based global router is proposed. The main characteristic of the proposed global router is that it uses a fairly accurate congestion map to implement congestion-based hierarchy in the design, to reduce the solution time of the ILP solver. This strategy also allows for special attention in dealing with congestion in the most problematic areas. The flow chart specifying the operation of the proposed router is shown in Figure 2. The proposed router starts with decomposing nets into two-pin subnets and creates initial minimum length trees for each subnet. Using these initial trees, it is proposed to generate a congestion map using the integer relaxation model proposed in Section 3.2. Then, to reduce the size of the problem, it is proposed to construct several levels of hierarchy based on the predicted congestion in the circuit. The routing problem in each level of the hierarchy is solved until, at the final level, every edge in the routing grid has been considered. The major steps of the proposed flow shown in Figure 2 are discussed in detail in the following sections.

3.1. Net Decomposition and Initial Tree Generation. The first step taken by the proposed router is to decompose nets into two-pin subnets using a rectilinear Steiner tree (RST) produced by FLUTE [21]. The two points, each being either a pin or a Steiner point, connecting each branch in the RST become a subnet. The proposed global router produces a

single tree for collinear subnets and two L-shaped trees for noncollinear subnets. This is the same first step taken by other ILP-based routers [15, 31, 32]. Net decomposition can dramatically reduce the complexity of tree generation, since only two pins need to be considered for each subnet versus a potentially large number of pins for a multipin net.

3.2. Congestion Map Generation. In this section, the proposed method for generating a congestion map is described. The major property of the proposed congestion map is that it is based on an actual routing solution; however, it can still be calculated quickly using a special unimodular ILP relaxation. In addition, the congestion map generation is flexible in that any linear objective can be used. In the rest of this section, the relevant mathematics followed by the proposed congestion map generation technique is discussed.

3.2.1. Unimodular Integer Linear Programs. An important property of linear programs is that a vertex in the constraint polyhedron will yield the optimal value. This was the fundamental observation that lead to the development of the vertex-following Simplex algorithm for solving Linear Programs (LPs) [33]. This idea has further ramifications for ILPs: if the constraint matrix is unimodular, the solution to the relaxed LP will be integer valued. This property is used in this paper to produce congestion maps.

Definition 1 (Unimodularity). A square matrix with integer entries is unimodular if its determinant is 1, −1, or 0.

An immediate consequence of this is that for square systems $\mathbf{Ux} = \mathbf{b}$ with $\mathbf{b}$ made up of integers and $\mathbf{U}$ is unimodular, if $\det(\mathbf{U}) \neq 0$, the solution for $\mathbf{x}$ will also be integer. This follows from Cramer's Rule for obtaining the solution to linear systems using determinants.

Definition 2 (Total unimodularity). A rectangular matrix with integer entries is called totally unimodular if all of its square submatrices are unimodular.

3.2.2. Integer Relaxation Congestion Maps. To produce unimodular ILP formulation, the edge capacity constraints are removed from the problem. The remaining choice constraints ($\sum_{t_{i,j} \in T_i} x_{i,j} = 1,\ \forall n_i \in N$) give the constraint matrix a totally unimodular structure, as mentioned in [27].

Having a totally unimodular ILP means that the integer integrality constraints ($x_{i,j} \in \{0,1\},\ \forall n_i \in N,\ \forall t_{i,j} \in T_i$) can be relaxed to linear constraints resulting in an LP problem, while the optimal solutions of the ILP and LP can be the same. This eliminates the need to use a costly branch-and-bound scheme and avoids inaccuracies due to rounding to obtain integer solutions. The integer relaxation of the global routing problem can be stated as

$$\begin{aligned} \text{(Integer Relaxation)} \quad \text{minimize} \quad & \mathbf{w}'\mathbf{x}, \\ \text{subject to} \quad & \mathbf{Ux} = \mathbf{1}, \\ & \mathbf{0} \leq \mathbf{x}, \end{aligned} \tag{2}$$

where $\mathbf{w}$ contains weights associated with each tree in $\mathbf{t}$ and $\mathbf{x}$ represent the binary selection vector for the trees produced for the problem. Matrix $\mathbf{U}$ is a unimodular matrix representing the exclusivity constraints in which only one tree can be chosen for each net. The $\mathbf{x} \leq \mathbf{1}$ constraints are removed because they are implied through the constraints $\mathbf{Ux} = \mathbf{1}$. In addition to obtaining binary solutions through solving this LP problem, this model can be evaluated more quickly than other models because of the reduced number of constraints, which are roughly cut in half. The drawback is that the capacity constraints are not considered and overflow can occur. This problem is to some extent remedied by proposing a modified objective function for the problem in (2) that incorporates a congestion measure in the objective function weights, $\mathbf{w}$. In the rest of this section, the proposed congestion weights are discussed.

In [28], an ILP-based tree congestion measure, *routing demand*, $\mathrm{rd}(\cdot)$, is proposed which is related to the total congestion encountered by a tree. In this model first, a predicted edge demand, $d_{\mathrm{pre}}(e_k)$, is calculated for each edge, e_k, as follows:

$$d_{\mathrm{pre}}(e_k) = \sum_{\{t_{i,j} \mid e_k \in t_{i,j}\}} P\left(t_{i,j}\right), \tag{3}$$

where $P(t_{i,j})$ is the probability for the tree $t_{i,j}$ to be selected in the final solution. This probability is set to be equal to the inverse of the number of trees generated for net i:

$$P\left(t_{i,j}\right) = \frac{1}{|T_i|}. \tag{4}$$

The routing demand value for a tree, $\mathrm{rd}(t_{i,j})$, proposed in [28] is equal to the sum of the predicted edge demands, $d_{\mathrm{pre}}(\cdot)$, of the edges that the tree contains:

$$\mathrm{rd}\left(t_{i,j}\right) = \sum_{e_k \in t_{i,j}} d_{\mathrm{pre}}(e_k). \tag{5}$$

This calculation tries to foretell the amount of congestion that the tree will be a part of but does not penalize the overflows.

In this paper, a new method to predict the congestion encountered in the path of a tree is proposed that tries to remedy the problem in [28]. This technique focuses on the available edge supply which is equal to the edge capacity less demand for an edge, to include consideration of edge capacities and overflow. The demand in this case is predicted by using (3). However, nets with only a single tree are assumed routed and their edge demand is subtracted from the initial edge capacities. In addition, there is a distinct jump in the calculations of the edge routing supply when available resources become negative; that is, the edge is predicted to overflow. The routing supply of an edge, $\mathrm{rs}_e(e_k)$ is calculated using the following formula:

$$\mathrm{rs}_e(e_k) = \begin{cases} c_{e_k} - d_{\mathrm{pre}}(e_k) & \text{if } c_{e_k} - d_{\mathrm{pre}}(e_k) \geq 0, \\ \gamma_{\mathrm{pnlty}} \times \left(c_{e_k} - d_{\mathrm{pre}}(e_k)\right) & \text{if } c_{e_k} - d_{\mathrm{pre}}(e_k) < 0, \end{cases} \tag{6}$$

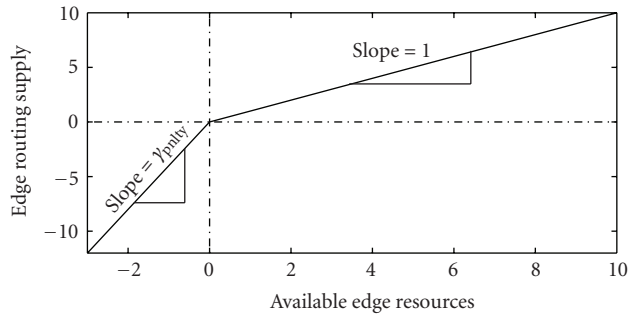


FIGURE 3: The routing supply of an edge.

where c_{e_k} and $d_{\text{pre}}(e_k)$ are the capacity and the demand of edge e_k, respectively. The demand of an edge $d_{\text{pre}}(e_k)$ is calculated using (3). $\gamma_{\text{pnlty}} > 1$ is the penalty factor associated with overflowing edges. The top equation in (6) shows the routing supply when an edge is not overflown and the bottom equation is the case when the edge is overflowing.

In Figure 3, the relation between the routing supply and the available edge resources is shown. In this figure the horizontal axis shows the available resources and the vertical axis shows the value assigned as the edge supply $\text{rs}_e(e_k)$. The slope in the region where there are routing resources available, positive supply, is 1. When there is no more resources available, an excess of demand, the slope is $\gamma_{\text{pnlty}} > 1$ (set to one quarter of the edge capacity in the implementation) to more heavily penalize predicted overflow.

Once the routing supply for each edge is considered, then the routing supply for a tree $\text{rs}_t(t_{i,j})$ needs to be estimated as the summation of the routing supplies of the edges that it passes through:

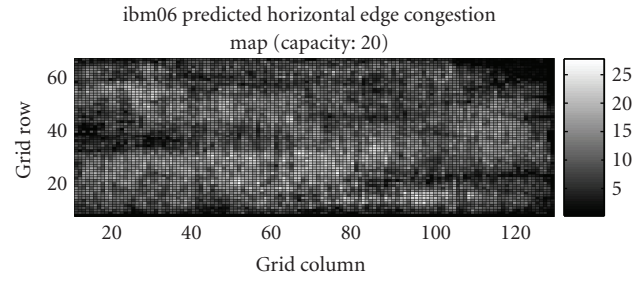
$$\text{rs}_t\left(t_{i,j}\right) = \sum_{e_k \in t_{i,j}} \text{rs}_e(e_k). \tag{7}$$

The routing supplies obtained as mentioned above are then incorporated as weights $w_{i,j}$ for trees in the objective of (2), where

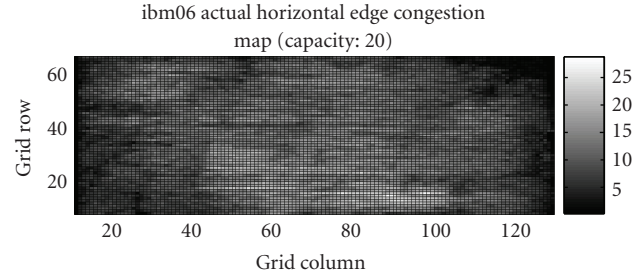
$$w_{i,j} = -\text{rs}_t\left(t_{i,j}\right). \tag{8}$$

Since it is desirable to maximize routing supply, that is, minimize negative routing supply, then, (2) which is referred to as the routing supply integer relaxation (RSIR) is solved by using the Simplex method (CPLEX 9.0 library [34]). The Simplex method guarantees to find a vertex solution, which will be integer, as opposed to an interior-point method which may find an optimal solution in the interior of the optimal face, that is, noninteger solution.

It should be mentioned that for a long tree with both congested and noncongested edges, the routing supply can be high, but since at this stage only minimum length trees are produced for each net, the length of all trees for a net will be equal and the tree with the overall higher routing supply will be preferred in the final solution. In addition, the penalty $\gamma_{\text{pnlty}} > 1$ associated with the edges with negative supply will emphasize the congested edges.



(a) Predicted congestion

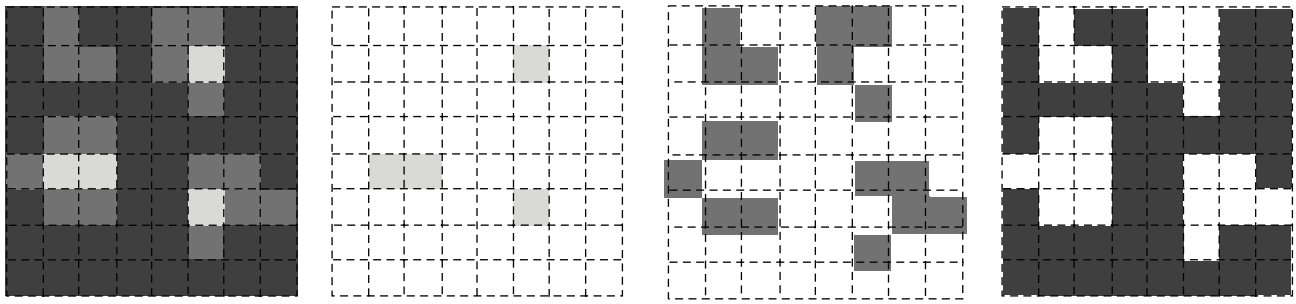


(b) Final solution congestion

FIGURE 4: A predicted congestion map and the actual congestion after routing for the horizontal edges in ibm06. Shading indicates edge demands.

An example of a map obtained from (2) is shown in Figure 4(a), where the predicted low congested areas are shown with a dark shade and high congested areas are shown in a bright shade. The figure illustrates that the congested areas are predicted to occur in several areas spread all over the design. The final solution after routing is shown in Figure 4(b). As can be seen, the most highly congested areas in the prediction remain amongst the most congested areas in the final solution. The maximum edge demand decreases from 28 in the predicted map to 21 in the actual map. This is a result of the router's ability to remove congestion to satisfy the edge capacity constraints which are not considered in the prediction. With the decrease in maximum edge demand in the final solution, there are more edges with demand near the maximum. This results in larger brightly shaded regions compared to the prediction map, although the congestion also exists in the prediction with a less bright shade. Further results indicating the accuracy of the congestion map are given in Section 4.1.

3.3. Hierarchy Construction by Utilizing a Congestion Map. Once a congestion map is constructed using (2), it is utilized as a guide for building a congestion-based hierarchy. The hierarchy is structured to first carefully consider the most congested edges and then consider successively less congested edges until routing is completed. To the best of our knowledge, this is the first router to use individual routing grid edges to organize a hierarchal structure. Using routing edges to define the hierarchy is advantageous in that the most problematic, congested edges are dealt with directly as opposed to region-based hierarchies. An example of how



(a) Congestion map (b) Edges considered in the first level (c) Edges considered in the second level (d) Edges considered in the third level

FIGURE 5: An illustration of the congested edge hierarchy created from a congestion map used by the proposed router. The routing grid is shown by dashed lines and selected edges in each level are shown by shading. In the congestion map, congestion is predicted to be higher as the shade gets brighter.

the hierarchy is formed based on the predicted congestion is illustrated in Figure 5. The congestion map for a sample design is shown in Figure 5(a). Here, the more congested an edge is, the brighter the shade of it will be. The least congested edges are dark gray. Using the proposed hierarchy, the most congested edges, which are shown as the lightest gray in Figure 5(b), make up the set of edges considered in the first level of the hierarchy. The next most congested edges are considered in the second level and are indicated in Figure 5(c). Finally, all remaining edges are considered in the third and final level as shown in Figure 5(d).

Once the hierarchy is established, at each level, an ILP problem is formulated and solved to find routes for subnets with trees passing through the edges considered at the level. All trees for the selected subnets are considered during the routing even if only one tree contains edges for the current level. These subnets may be spatially separated and not localized to a small initial routing region. Once all the subnets belonging to one level have been routed, the routing proceeds with the set of the next most highly congested edges, which defines the second level in the hierarchy. The number of highly congested edges considered in each level also defines a trade-off between willingness to alleviate congestion and runtime. In this paper, it is proposed that in the highest/most congested levels, a small percentage of edges are considered, and in lower/less congested levels the remaining edges in the circuit are considered. This is based on the observation that there are several congested regions in the congestion maps for each circuit and usually a few large areas of relatively uncongested regions. From the RSIR congestion maps produced for the experimental circuits, only 6% of the edges are expected to have overflow on average. The proposed router is to focus on the edges that are expected to overflow and then consider everything at once after the most heavily congested areas are dealt with. The hierarchical strategy proposed here is implemented in the context of ILP-based routing but can be directly applied to other concurrent methods.

3.4. A Single Level ILP Solution Methodology. In this section, the methodology proposed to solve a single-level of hierarchy is explained. The process flow for routing a single level is illustrated in Figure 6.

At the beginning of the single level ILP, each subnet has one or two minimum length, minimum bend trees. The routing supply of each one of these trees is calculated using (7). If the routing supply of a tree is high, meaning it contains uncongested edges, and no congested edges exist in the path of the tree, then that tree is chosen for the routing solution. The minimum threshold for this prerouting of nets is set high (set to three quarters of the sum of the edge capacities of the edges in the tree) to be conservative but still able to remove some variables from the formulation. For the subnets that do not have a tree with a high routing supply, an additional tree is generated.

After the additional trees are generated, the routing problem is formulated as an ILP which includes capacity constraints. Since there can be no guarantee that the ILP discussed in (1) can have a feasible solution, a variation of the the ILP where the objective is the maximization of the number of routed nets [15, 16, 31] is used. In the maximization of the routed nets model, nets are routed without creating overflow. The formulation is given in the following equation:

$$\begin{aligned} \text{maximize} \quad & \sum x_{i,j} \\ \text{subject to} \quad & \sum_{t_{i,j} \in T_i} x_{i,j} \leq 1, \quad \forall n_i \in N, \\ & \sum_{\{i,j \mid e_k \in t_{i,j}\}} x_{i,j} \leq c_{e_k}, \quad \forall e_k \in E, \\ & x_{i,j} \in \{0,1\}, \quad \forall n_i \in N, \ \forall t_{i,j} \in T_i. \end{aligned} \tag{9}$$

This is basically the same equation as stated in (1) with the objective changed from minimization to the maximization of all routed subnets. In addition, the first constraint set has been changed from the generic ILP routing formulation in (1) to allow some net not to be routed.

To enhance the quality of the solution of (9), in this paper it is proposed to add a weight for each tree in its objective and

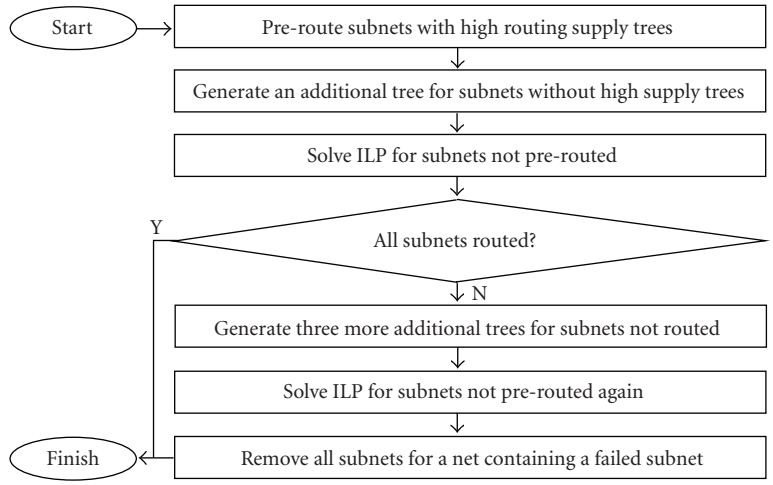


Figure 6: The flow chart specifying the proposed single-level routing process.

maximize a weighted sum of trees considered at each level. Therefore, the objective problem of (9) is changed to

$$\text{maximize} \quad \sum w_{i,j} x_{i,j}, \tag{10}$$

where $w_{i,j}$ are the weights that show how desirable a tree is. The calculated routing supplies for trees can be a good measure to show this desirability. However, using the raw values of the $\text{rs}_t(t_{i,j})$ can result in suboptimal solutions, since it is required to ensure that the maximum number of nets is routed. If the routing supplies for the trees of a net are very different, then suboptimal trees can be chosen.

An example of the problem that can arise if the routing supplies of trees for a net are very different in range is illustrated in Figure 7. In this example there are three nets and each has two trees. The routing supply $\text{rs}_t(\cdot)$ is given for each tree. Here, for simplicity each subnet has two trees, the most preferred one has a routing supply equal to 1, and the less preferred one has a routing supply of 0.2. But because of previous levels in the hierarchy the edges that contain $t_{2,1}$ and $t_{3,1}$ are overflowing. Although those two trees have the highest routing supply, they cannot be used due to the overflow; so only $t_{2,2}$ and $t_{3,2}$ may be routed. However, because $t_{1,1}$ has a higher routing supply ($= 1$) than the sum of the routing supplies for $t_{1,2}$, $t_{2,2}$ and $t_{3,2}$ ($= 0.6$), it will be routed. This makes the near capacity edges to reach their capacity; so neither $t_{2,2}$ nor $t_{3,2}$ can be routed and those two subnets fail to be routed. As a result, only one net is routed as opposed to three.

To prevent the problem shown in the example in Figure 7, the routing supply is normalized to a number between 0.99 and 1 through the following equation:

$$\text{rs}_{t\text{nrml}}\left(t_{i,j}\right) = 0.99 + 0.01 \times \frac{\text{rs}_t\left(t_{i,j}\right) - \text{rs}_{t\,\min,n_i}}{\text{rs}_{t\,\max,n_i} - \text{rs}_{t\,\min,n_i}}, \tag{11}$$

where $\text{rs}_{t\text{nrml}}(\cdot)$ is the proposed normalized routing supply of a tree $\text{rs}_{t\,\max,n_i}$ and $\text{rs}_{t\,\min,n_i}$ are the maximum and minimum routing supplies amongst the trees produced for net n_i. The

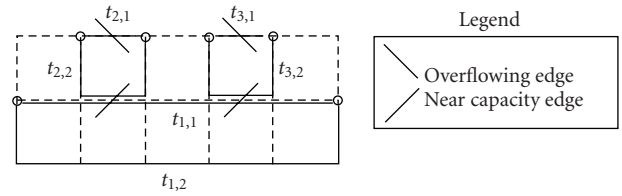


Figure 7: An example of how normalizing routing supply may fail if the normalization range is not close to 1.

normalized routing supply is set to 1 if $\text{rs}_{t\,\max,n_i} = \text{rs}_{t\,\min,n_i}$ for some n_i. With the minimum possible weight at 0.99, it is very unlikely for a situation like the one illustrated in Figure 7 to occur.

3.5. Multilevel Formulation and Solution Methodology. At each level of hierarchy an ILP formulation needs to be solved. This new formulation can be written in matrix format as follows:

$$\begin{aligned} \text{maximize} \quad & \mathbf{rs}_{t\text{nrml}}^{\text{lvl}}{}' \mathbf{x}^{\text{lvl}} \\ \text{subject to} \quad & \mathbf{U}^{\text{lvl}} \mathbf{x}^{\text{lvl}} \leq \mathbf{1}^{\text{lvl}}, \\ & \mathbf{A}^{\text{lvl}} \mathbf{x}^{\text{lvl}} \leq \mathbf{cap}^{\Sigma\text{lvl}}, \\ & \mathbf{x}^{\text{lvl}} \in \{0, 1\}, \end{aligned} \tag{12}$$

where the lvl superscript emphasizes that the problem dimensions have been scaled down to only include the subnets in the current level. $\mathbf{rs}_{t\text{nrml}}^{\text{lvl}}$ is the vector of routing supplies associated with each tree and is discussed in the previous section. Matrices $\mathbf{U}^{\text{lvl}}$ and $\mathbf{A}^{\text{lvl}}$ are the integrality and capacity constraint matrices, respectively. $\mathbf{1}^{\text{lvl}}$ is a vector of one of the appropriate size and $\mathbf{cap}^{\Sigma\text{lvl}}$ is the vector of the available capacities at the current level. This problem

Table 1: Comparison between the edge demand predicted by the congestion map, calculated after solving (12), and after maze routing for the proposed global router on the ISPD98 benchmark suite.

Circuit	Congestion map prediction			After solving (12)			After maze routing		
	Avg	Max	% OV edges	Avg (Δ%)	Max (Δ%)	% edges at max capacity	Avg (Δ%)	Max (Δ%)	% OV edges predicted
ibm01	7.2	30	8.9	6.5 (−9.6%)	14 (−53.3%)	3.6	8.7 (19.6%)	17 (−43.3%)	3.5
ibm02	16.0	51	9.5	14.8 (−7.7%)	34 (−33.3%)	1.4	17.8 (11.0%)	36 (−29.4%)	3.7
ibm03	14.0	44	6.1	13.7 (−2.7%)	30 (−31.8%)	3.4	14.6 (4.0%)	30 (−31.8%)	0.5
ibm04	13.0	37	8.1	12.2 (−5.9%)	23 (−37.8%)	2.7	14.3 (9.6%)	26 (−29.7%)	24.6
ibm05	24.7	65	0.1	24.7 (0.0%)	63 (−3.1%)	0.1	24.7 (0.0%)	63 (−3.1%)	0.0
ibm06	16.7	58	6.6	15.8 (−5.3%)	33 (−43.1%)	1.5	17.9 (7.4%)	33 (−43.1%)	0.8
ibm07	14.6	53	3.9	14.2 (−3.2%)	36 (−32.1%)	1.1	16.0 (9.4%)	36 (−32.1%)	3.7
ibm08	16.3	47	5.2	15.6 (−3.9%)	32 (−31.9%)	1.7	17.0 (4.8%)	33 (−29.8%)	1.2
ibm09	12.4	43	6.3	11.7 (−5.8%)	28 (−34.9%)	2.9	13.1 (5.5%)	28 (−34.9%)	0.2
ibm10	17.4	57	4.6	16.9 (−2.8%)	40 (−29.8%)	1.1	18.2 (4.7%)	41 (−28.1%)	5.4
Overall	15.2	48.5	5.9	14.6 (−4.7%)	33.3 (−33.1%)	1.9	16.2 (7.6%)	34.3 (−30.5%)	4.4

is solved directly without relaxing the binary constraints. Because the problem is solved hierarchically, the added runtime in solving the problem without relaxing it to a linear program is not significant and any rounding issues can be avoided.

Within each level, the ILP formulation is solved once after producing a single additional tree for each subnet and once again after producing three additional trees for each subnet if there were subnets that could not be routed the first time. The commercial solver CPLEX 9.0 [34] is used for solving the ILPs. CPLEX uses a branch-and-bound scheme to solve the ILPs directly.

4. Experimental Results for the Proposed Router

This section presents the results from using the proposed router on the ISPD98 benchmark suite [35]. The program is written in C and all experiments are performed using an UltraSPARC workstation running SunOS 5.9 with 4 GB of RAM. In the remainder of this section, an evaluation of the performance and a comparison of the proposed router with the other ILP-based routers are given.

4.1. Congestion Map Evaluation. To ensure that the congestion maps generated are a good representation of the final solution, the predicted demand of edges is compared with the actual edge demand after solving the multilevel ILP formulation of (12) and the final solution of the proposed router in Table 1. The final solution is obtained by maze routing nets that were unable to be routed using (12). To deal with the few nets that could not be routed by the proposed formulation, the Labyrinth maze router in [36] is used.

Information about the RSIR congestion map accuracy is presented in Table 1. The table is divided into three groups to show congestion results for the RSIR congestion map, the solution of (12), and the after maze routing solution. Each group has a column containing the average, "avg", and maximum, "max", edge demands. The average and maximum are also shown in percentage change, Δ%, with respect to the RSIR congestion prediction for the latter two groups. It can be seen that the predicted average edge demand is within 5% of the congestion after solving (12) and 7.6% from the after maze routing congestion. The predicted maximum edge demand is much higher, 33.1% and 30.5% respectively, than the maximum demand resulting from solving (12) and maze routing. This is due to the fact that the RSIR map does not consider the edge capacity constraints. The percentage of edges overflowing in the RSIR congestion map is shown in Column 4. On average, 5.9% of the edges are overflowing in the prediction. There are no edges overflowing after solving (12) as overflow does not occur. Column 7 shows the percentage of edges that have reached their capacity after solving (12). There is less than 1% of edges overflowing, on average, after maze routing (not shown in Table 1). The percentage of those edges that were predicted by the RSIR map to be overflowing is shown in the last column. Only 4.4% of the edges that overflow were overflowing in the prediction. This illustrates that the routing methodology is effective at removing overflow from the predicted overflowing edges.

4.2. Proposed ILP Router Performance

4.2.1. Building Hierarchy Levels. The proposed router uses a three level hierarchy with the first level considering the 2% most congested edges, the second level considering the 2% next most congested edges, while the third level considering all remaining edges. Having a small problem size in each of the first 2 levels allows the solution to be found quickly for each level. It also allows the subnets using the most congested edges to have the highest degree of freedom in finding alternate routes since they are not competing for resources with the majority of subnets that are in the third level. The percentage of subnets considered during each level

Table 2: Percentage of subnets considered during each level by the proposed router.

Circuit	% subnets level 1	% subnets level 2	% subnets level 3
ibm01	4.7	5.4	89.9
ibm02	2.1	3.1	94.8
ibm03	4.7	4.4	90.8
ibm04	3.7	3.7	92.6
ibm05	2.2	4.9	92.9
ibm06	3.2	2.6	94.2
ibm07	2.6	4.8	92.6
ibm08	3.2	2.7	94.1
ibm09	2.5	5.1	92.4
ibm10	1.9	3.1	95.0
Average	3.1	4.0	92.9

is given in Table 2. The percentage of subnets considered during the first, second, and third levels of the hierarchy is shown in Columns 2, 3, and 4, respectively. On average 3.1% of the subnets are attempted to be routed in the first level and 4.0% in the second level.

4.2.2. Solving One Level of ILP. During each level of the ILP problem, it was decided to use branch-bound versus solving the relaxed LP in order to reduce the overflow which can occur after rounding the LP nonbinary solutions to obtain the binary values. In Table 3, the results in terms of time to solve and the number of overflow when solving the three levels of hierarchy are given. The total runtime for the proposed router using a direct ILP solution of (12) or an LP relaxation of (12) is reported in Columns 2 and 3, respectively. The runtimes are comparable for all benchmarks with only a slight increase when solving the ILP versus the LP relaxation. The amount of subnets requiring their tree variables to be rounded to obtain integer solutions is listed in Column 4. On average 3.0% of the subnets require rounding for the LP relaxation. This rounding creates substantial overflow, as shown in Column 5.

Another issue with the runtime is the time spent on producing additional trees. The number of trees that get produced for each subnet defines a balance between the willingness to prevent overflow and the runtime. In Table 4, the total time for solving (9) using only one additional tree per subnet, Column 2, and the percentage change in runtime when 4 additional trees are generated, Column 3, are given. It can be seen that on average, there is 243% increase in runtime. This runtime increase is due to the increase in time spent generating trees as well as the increase in time to solve the ILP problem with more variables. In Columns 4 and 5, the percentage of the total time taken by the router solving 7 to produce one additional tree and four additional trees is shown. These data show that the percentage of time spent producing trees also increases.

4.2.3. Overall Performance of the Proposed Router. The results from running the proposed router are shown in Table 5. The total wire length and vias for each benchmark are reported in Columns 2 and 3, respectively. Since the router maximizes the number of routed nets without creating overflow, the

Table 3: Solution times and rounding results on the ISPD98 benchmark suite using (12) or a linear relaxation of (12).

	Direct ILP	LP Relaxation		
Circuit	Time (s)	Time (s)	% rounded	OV from rounding
ibm01	4	4	3.6	69
ibm02	20	21	3.6	188
ibm03	41	40	6.0	258
ibm04	35	33	4.5	321
ibm05	37	35	0.0	0
ibm06	54	49	3.1	276
ibm07	168	148	2.4	231
ibm08	173	160	2.5	216
ibm09	185	172	2.8	292
ibm10	343	308	2.0	269
Average	106	97	3.0	212

Table 4: Additional tree generation runtime information for the model in (9).

Circuit	Total time 1 add. tree (s)	% change in total time 4 add. trees	% time gen. trees 1 add. tree	% time gen. trees 4 add. trees
ibm01	11.0	255.0%	36.4%	53.6%
ibm02	29.0	203.0%	41.4%	57.6%
ibm03	28.0	261.0%	46.4%	57.5%
ibm04	44.0	289.0%	54.5%	56.7%
ibm05	37.0	132.0%	13.5%	30.6%
ibm06	54.0	261.0%	42.6%	53.2%
ibm07	105.0	193.0%	44.8%	57.1%
ibm08	135.0	244.0%	56.3%	67.2%
ibm09	236.0	304.0%	58.5%	74.8%
ibm10	252.0	291.0%	57.9%	73.0%
Average	—	243.3%	45.2%	58.1%

Table 5: The proposed router's results using (12) on the ISPD98 benchmark suite.

Circuit	WL	Vias	Total time (s)	Unrouted nets
ibm01	51832	11542	4	302 (3.3%)
ibm02	149936	26146	20	420 (2.9%)
ibm03	139562	21581	41	149 (1.0%)
ibm04	151236	24819	35	376 (1.9%)
ibm05	410120	50619	37	0 (0.0%)
ibm06	255351	39155	54	269 (1.0%)
ibm07	349630	53481	168	325 (1.0%)
ibm08	382907	65656	173	275 (0.8%)
ibm09	379736	59005	185	603 (1.5%)
ibm10	550287	90621	343	413 (0.8%)

Table 6: The proposed router + maze router's results and comparisons on the ISPD98 benchmark suite.

Circuit	Total wire length					Number of vias					Total overflow					Total runtime (s)				
	Prop.	BR	SW	MO	FGR	Prop.	BR	SW	MO	FGR	Prop.	BR	SW	MO	FGR	Prop.	BR	SW	MO	FGR
		%Δ	%Δ	%Δ	%Δ		%Δ	%Δ	%Δ	%Δ		Δ	Δ	Δ	Δ		%Δ	%Δ	%Δ	%Δ
ibm01	70112	6.9	6.1	0.1	10.8	17262	11.8	14.4	**−8.6**	0.8	87	**−15**	**−168**	**−71**	87	9	**−59.1**	**−99.9**	**−80.4**	**−75.7**
ibm02	178910	0.1	2.8	0.7	4.9	35698	9.7	16.4	**−1.1**	**−5.9**	10	**−23**	2	**−1**	10	27	**−76.5**	**−99.5**	**−59.7**	**−55.0**
ibm03	150051	**−0.8**	1.7	**−0.3**	1.5	25186	**−2.1**	10.4	**−0.6**	**−21.3**	3	3	3	0	3	43	2.4	**−99.2**	13.2	168.8
ibm04	176264	1.7	2.1	0.0	4.8	34183	10.9	19.5	**−0.6**	**−11.1**	360	51	**−258**	**−87**	360	41	**−50.0**	**−99.7**	**−58.6**	**−68.9**
ibm05	410120	0.1	0.1	0.0	**−1.1**	50619	**−1.2**	0.6	**0.3**	**−34.3**	0	0	0	0	0	37	**−76.0**	**−38.3**	32.1	48.0
ibm06	284724	0.8	1.7	0.0	1.3	48983	7.2	14.3	**−0.4**	**−14.1**	3	3	3	3	3	60	**−33.3**	**−99.2**	30.4	**−7.7**
ibm07	388707	2.6	1.8	0.4	5.2	63653	4.6	11.9	**−3.2**	**−19.0**	20	**−33**	20	11	20	179	16.2	**−97.0**	6.5	126.6
ibm08	416765	0.4	0.8	0.1	1.6	76214	1.2	9.9	**−1.0**	**−18.8**	7	7	7	**−8**	7	182	**−18.4**	**−97.5**	16.7	208.5
ibm09	424643	1.4	1.9	**−0.2**	1.8	72722	5.8	12.9	**−1.8**	**−16.1**	17	17	17	0	17	211	17.2	**−98.7**	44.5	193.1
ibm10	595416	0.4	0.7	0.0	2.1	104782	4.2	9.9	**−1.8**	**−18.2**	38	38	38	**−13**	38	362	47.8	**−94.1**	73.2	17.2
Average	—	1.4	2.0	0.1	3.3	—	5.2	12.0	**−1.9**	**−15.8**	—	4.8	**−33.6**	**−16.6**	54.5	—	**−23.0**	**−92.3**	1.8	55.5

total overflow is zero and is not reported in this table. The total runtime is reported in seconds in Column 4. The number of nets that were not able to be routed without creating overflow is listed in Column 5. These values range from 0 for ibm05 up to 603 for ibm09. The unrouted nets are likely a result of the limitations of the additional trees that are generated by the algorithm. Since each additional tree has only a small detour from existing trees, there are situations where all of the detoured trees still contain at least one highly congested edge.

The results for the combined proposed router and maze routing the unrouted nets are shown in Table 6. All of the values have increased by adding the maze-routed nets and are shown in columns with the heading Prop. The via count increases quite dramatically for the small number of nets that have been added. This is in part due to the nature of the maze router attempting to minimize the added wirelength and paying less attention to the additional vias. The large increase is also due to the fact that most of the edge resources have been used up after the proposed router terminates. Therefore, paths with high number of bends may be all that are possible at this stage. The overflow, shown in Column 12, has also increased from 0 for most benchmarks. This shows that some of the routes could not be completed without violating some of the edge capacity constraints. The maximum overflow is 360 for ibm04. This small addition of overflow is usually handled by the detailed router [15].

4.3. Comparisons with Other Modern Academic Routers. The proposed router with maze routing is compared with several modern academic routers in this section. The three modern ILP-based routers are first compared and a modern academic router that does not use ILP formulations will be compared to at last. The comparisons in Table 6 are denoted by the Δ symbol which denotes the percentage change between the proposed and the stated router. Negative values mean that the proposed router yields a lower value than the router it is being compared with. These negative values are shown in bold.

The first ILP-based router that is compared to the proposed router is BoxRouter [31]. BoxRouter uses a progressive ILP-formulation that begins by routing inside of a box encompassing the most congested area of a circuit. The proposed router uses 1.4% more wirelength than BoxRouter and 5.2% more vias. It also has less instances of overflow on three of the benchmarks and on average 4.8 more instances of overflow, but it is able to route each circuit in 22.9% less runtime than BoxRouter. The two routers are well matched. Because BoxRouter is able to perform maze routing at each level it is less likely to have to take large detours with high bends for subnets that could not be routed in each level. This contributes to the lower via count and wirelength compared to the proposed router with maze routing.

Sidewinder [15] is an ILP-based router that makes use of a dynamic congestion map to find trees each iteration it performs routing. The results using Sidewinder were gathered using CPLEX 10.1 and taken from [15] which used an AMD Opteron 2.4 GHz machine with 4 GB of RAM. The total runtime is also taken from [15] since the router is not publicly available.

Sidewinder uses 2.0% less wirelength and 12.0% less vias than the proposed router with maze routing. As can be seen, the proposed router has less total overflow than Sidewinder on average but Sidewinder has zero overflow for seven of the ten benchmarks. The cost of having less wirelength and vias with more benchmarks having no overflow is a very significant increase in runtime. Although the experiments are on different machines, the machines are of the same generation and Sidewinder uses a newer version of CPLEX than the proposed router. Discrepancies in the machines cannot account for the 92.3% reduction in runtime taken by the proposed router. Since Sidewinder maintains a dynamic

congestion map and generates a significant amount of trees that never get used, it has a very high runtime cost. The proposed router only generates a single congestion map very quickly at the start which is accurate enough to never have to update it again.

The final modern ILP-based router that is compared to the proposed router is Multiobjective Router [16] using its maximization of routed nets model. The results for Multiobjective Router are taken from [16] which used CPLEX 10.0. The runtimes are scaled via the Labyrinth router's runtimes which are reported in [16] and were also ran on the proposed routers' machine. The proposed router reduces the number of vias and the number of instances of overflow significantly while only taking 0.1% more wirelength and 2.1% more runtime, on average. Multiobjective Router does not use hierarchy or a congestion map for routing. Without foreseeing where overflow is likely to occur, the router ends up being less able to avoid congested regions compared to the proposed router. In the proposed router, the congestion-based hierarchy is better able to alleviate the congestion in the most congested areas.

FGR [11] is a router that uses iterative RRR with congestion information computed with Lagrange multipliers for congestion estimation. It does not make use of ILP-based routing techniques. FGR is able to use 3.3% less wirelength and has no overflow for each benchmark at the expense of a significant increase in the number of vias. The runtime for FGR is less than the proposed router in six of the ten test cases. The amount of vias impacts the manufacturability of a circuit and increases delays. Given that detailed routers are able to handle small amounts of overflow [15], the solutions produced by each router have different merits.

Overall, the proposed router is able to compete with several state-of-the-art global routers. The router has advantages and disadvantages in terms of the metric being compared with each of the other modern academic routers. The proposed router is faster than BoxRouter and Sidewinder, uses fewer vias than Multiobjective Router and FGR, and has less overflow than Sidewinder and Multiobjective Router when considering averages. These results show that the proposed router is of high quality and is preferred to other routers with respect to different metrics.

5. Summary

In this paper a new multilevel ILP-based global router is proposed. The router is hierarchical in nature and utilizes a congestion map calculated before routing to identify congested edges. These congested edges form a hierarchy in the router to decide which subnets get routed within each level. Each level in the hierarchy gives rise to an ILP formulation that maximizes the number of subnets routed. The proposed router was shown to be competitive with other modern academic routers.

The future work for this paper includes: improving the additional tree generation technique based on initial congestion maps, extending the router to handle 3D designs, and iteratively improving the congestion map using Lagrangian relaxation factors.

Acknowledgment

This work was partially funded by NSERC, Alberta Ingenuity Fund, and iCORE.

References

[1] J. Cong, "Challenges and opportunities for design innovations in nanometer technologies," in *Frontiers in Semiconductor Research: A Collection of SRC Working Papers*, pp. 54–57, Semiconductor Research, San Jose, Calif, USA, 1997.

[2] "International technology roadmap for semiconductors report," November 2007, http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[3] "ISPD2007 Global Routing Contest," June 2008, http://www.sigda.org/ispd2007.

[4] "ISPD2008 Global Routing Contest," June 2008, http://www.ispd.cc/contests.

[5] J. Lou, S. Thakur, S. Krishnamoorthy, and H. S. Sheng, "Estimating routing congestion using probabilistic analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 1, pp. 32–41, 2002.

[6] S.-P. Lin and Y.-W. Chang, "A novel framework for multilevel routing considering routability and performance," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '02)*, pp. 44–50, ACM, New York, NY, USA, 2002.

[7] R. T. Hadsell and P. H. Madden, "Improved global routing through congestion estimation," in *Proceedings of the 40th Conference on Design Automation (DAC '03)*, pp. 28–31, ACM, New York, NY, USA, 2003.

[8] M. Cho and D. Z. Pan, "BoxRouter: a new global router based on box expansion and progressive ILP," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 12, pp. 2130–2143, 2007.

[9] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic congestion prediction," in *Proceedings of the International Symposium on Physical Design (ISPD '04)*, pp. 204–209, Phoenix, Ariz, USA, 2004.

[10] A. B. Kahng and X. Xu, "Accurate pseudo-constructive wirelength and congestion estimation," in *Proceedings of the International Workshop on System Level Interconnect Prediction (SLIP '03)*, pp. 61–68, 2003.

[11] J. A. Roy and I. L. Markov, "High-performance routing at the nanometer scale," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '07)*, pp. 496–502, IEEE/ACM, Piscataway, NJ, USA, 2007.

[12] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang, "NTHU-route 2.0: a fast and stable global router," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '08)*, pp. 338–343, November 2008.

[13] C.-H. Hsu, H.-Y. Chen, and Y.-W. Chang, "Multi-layer global routing considering via and wire capacities," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '08)*, pp. 350–355, November 2008.

[14] Y. Zhang, Y. Xu, and C. Chu, "FastRoute3.0: a fast and high quality global router based on virtual capacity," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '08)*, pp. 344–349, November 2008.

[15] J. Hu, J. A. Roy, and I. L. Markov, "Sidewinder—a scalable ILP-based router," in *Proceedings of the International Workshop on System Level Interconnect Prediction (SLIP '08)*, pp. 73–79, ACM, New York, NY, USA, 2008.

[16] Z. Yang, S. Areibi, and A. Vannelli, "An ILP based hierarchical global routing approach for VLSI ASIC design," *Journal of Optimization Letters*, vol. 1, no. 3, pp. 281–297, 2007.

[17] T.-H. Lee and T.-C. Wang, "Congestion-constrained layer assignment for via minimization in global routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1643–1656, 2008.

[18] R. C. Prim, "Shortest connection networks and some generalization," *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[19] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*, North-Holland, Amsterdam, The Netherlands, 1992.

[20] M. R. Garey and D. S. Johnson, "The rectilinear steiner tree problem is NP-complete," *SIAM Journal of Applied Mathematics*, vol. 32, pp. 826–834, 1977.

[21] C. Chu and Y.-C. Wong, "FLUTE: fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 70–83, 2008.

[22] M. Pan and C. Chu, "FastRoute: a step to integrate global routing into placement," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '06)*, pp. 464–471, 2006.

[23] L. McMurchie and C. Ebeling, "Pathfinder: a negotiation-based performance-driven router for FPGAs," in *Proceedings of the International Symposium on Field Programmable Gate Arrays (FPGA '95)*, pp. 111–117, Monterey, Calif, USA, 1995.

[24] W. A. Dees Jr. and R. J. Smith II, "Performance of interconnection rip-up and reroute strategies," in *Proceedings of the 18th Design Automation Conference (DAC '82)*, pp. 382–390, IEEE Press, Piscataway, NJ, USA, 1981.

[25] D. Jansen, *The Electronic Design Automation Handbook*, Springer, New York, NY, USA, 2003.

[26] A. Vannelli, V. H. Quintana, and L. Vargas, "Interior point optimization methods: theory, implementations and engineering applications," *Canadian Journal of Electrical and Computer Engineering*, vol. 17, no. 2, pp. 84–94, 1992.

[27] T. Lengauer and M. Lügering, "Provably good global routing of integrated circuits," *SIAM Journal on Optimization*, vol. 11, no. 1, pp. 1–30, 2001.

[28] L. Behjat, A. Vannelli, and W. Rosehart, "Integer linear programming models for global routing," *INFORMS Journal on Computing*, vol. 18, no. 2, pp. 137–150, 2006.

[29] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 2, no. 4, pp. 223–234, 1983.

[30] J. Cong, M. Xie, and Y. Zhang, "An enhanced multilevel routing system," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '02)*, pp. 51–53, ACM, New York, NY, USA, 2002.

[31] M. Cho and D. Z. Pan, "BoxRouter: a new global router based on box expansion and progressive ILP," in *Proceedings of the Design Automation Conference (DAC '06)*, pp. 373–378, ACM, New York, NY, USA, 2006.

[32] Z. Yang, S. Areibi, and A. Vannelli, "A comparison of ILP based global routing models for VLSI ASIC design," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems (MWSCAS '07)*, pp. 1141–1144, 2007.

[33] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, Mass, USA, 1995.

[34] ILOG, "CPLEX: high-performance software for mathematical programming and optimization," http://www.ilog.com/products/cplex.

[35] "ISPD98 global routing benchmark suite," http://www.ece.ucsb.edu/~kastner/labyrinth.

[36] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 7, pp. 777–790, 2002.