A Modified Edge Removal Steiner Tree Heuristic for Global Routing in VLSI Design

Apichat Terapasirdsin\textsuperscript{1}, William F. Rotach\textsuperscript{2}, Naruemon Wattanapongsakorn\textsuperscript{3}, and Patrick H. Madden\textsuperscript{4}, Non-members

ABSTRACT

Routing is a key stage for VLSI physical design. Steiner tree construction is a well studied topic in design automation. There have been a number of significant theoretical advances in the past few years. The focus of this paper is on combining the speed and solution quality of a high quality Steiner heuristic with the reality of modern routing. Practical designs contain routing congestion and blockages; routing is implemented across multiple layers. Each routing layer has preferred directions, and connecting vias have significant cost. In a modern design, many trees are in competition with each other for scarce routing resources. The objective is not to simply build trees with the lowest length; they must also be low cost. We present an approach that is as fast as spanning tree construction, while accurately modeling routing costs.

Our work extends an earlier Steiner tree heuristic algorithm by adding the ability to minimize the routing congestion without altering the computational complexity of the underlying algorithm. We compare our CAST algorithm with the capacitated minimum spanning tree (CMST) and the ER algorithm finding that our approach offers impressive reduction in congestion cost in average about 23.1\% and 63.4\%, respectively.

Keywords: Steiner tree, Congestion, Global Routing, Rectilinear Distance

1. INTRODUCTION

Interconnect topology is a well studied area in integrated circuit design. As transistors have become faster, the interconnect wires have taken an increasing share of system delay \cite{1}. The wiring is now a limiting factor in the performance of advanced designs. Interconnect capacitance accounts for a great deal of switching power, and detours insertion to avoid routing congestion can introduce additional delay. Extreme cases of routing congestion can even prevent feasible physical design. As a result, additional space must be inserted to obtain a correct solution, causing larger size of the circuit design.

Much of the study of interconnect has focused on relatively simple metrics. “Manhattan distance,” in particular, has received a great deal of attention. While the metric is a reasonable approximation of interconnect cost, it fails to capture a number of important elements. Circuit interconnect almost always has “preferred direction,” with some layer alternating directions \cite{2}. Connecting the layers together are obtained with vias, which have high cost, and comparatively giving high failure rates; these are not modeled at all in traditional planar formulations. Further, the Manhattan distance metric does not capture routing congestion. So, when constructing an interconnect topology, one may wish to choose tree edges to avoid congested areas.

In this paper, we adapt the well known edge-removal Steiner tree heuristic \cite{3} to handle routing congestion by using an efficient heuristic routing cost caching approach, so that we maintain low run times, while accurately capturing congestion cost. We demonstrate our approach by applying the heuristic approach to various selected nets on ibm01 (ISPD98 IBM Benchmark).

The remainder of this paper is organized as follows. First, we briefly describe routing metrics, and some of the leading methods for generating interconnect topologies. We then describe routing congestion, and provide a simple but effective way to determine the routing cost between any two arbitrary points. Following this, we then describe the integration of routing cost calculations with the edge-removal heuristic, and then present experimental results. We conclude the paper with a summary of our research findings.

2. PREVIOUS WORK

There is an abundance of prior work on the Steiner problem. We will briefly touch on some relevant works as follows.

\textsuperscript{1,3} The authors are with The Department of Computer Engineering, King Mongkut’s University of Technology Thonburi 126 Pracha-Uthid Road, Bangmod, Tungkru, Bangkok, 10140, Thailand, Tel: +66(08)-37975555 , E-mail: apichatter@yahoo.com\textsuperscript{1}, s5510111@st.kmutt.ac.th\textsuperscript{1}, naruemon@cpe.kmutt.ac.th\textsuperscript{3}

\textsuperscript{2,4} The authors are with The Department of Computer Science, State University of New York at Binghamton, New York, USA., E-mail:
2.1 Manhattan Distance and the Steiner Problem

Manhattan (or “rectilinear”) distance minimization is the most common objective when generating interconnect topologies for circuit design. It is easy to compute, and roughly captures the spirit of the problem. Chiang [4] proposed an algorithm for obtaining a weighted (rectilinear) Steiner tree in the plane. The proposed global router at each step finds a weighted Steiner tree for a net, where weight of a region represents its “complexity”. This Weight of the region is dynamically changing.

Generating low cost spanning trees is trivial. An example of well-known algorithms is presented by Prim [5] and Kruskal [6]. The interconnect problem becomes interesting, when we include the possibility of introducing Steiner points. Hanan [7] showed that for \( v \) vertices, there were \( O(v^2) \) possible Steiner points for the Manhattan distance metric. While the added freedom of additional points allowing for tree length reductions is considered (of about 11% on random graphs), The problem becomes intractable.

Using computational geometry techniques, Kahng and Robins [8] developed an effective approach that iteratively inserted a single Steiner point at a time, seeking the best improvement at each step. By limiting the number of Steiner points that must be considered, and with efficient data structures, the approach produced excellent results with reasonable run times. There were further advances, with the GeoSteiner [9] approach from Wame, Winter, and Zachariasen, combining computational geometry and linear programming to obtain optimal solutions for surprisingly large graphs.

More recently, Ozdal [10] proposed an iterative rip-up and reroute (RNR) global routing algorithm that guides the routing iterations out of local optima through effective usage of congestion histories. Chu presented “FLUTE [11],” a surprisingly fast Steiner algorithm that is able to find optimal solutions for relatively small problems through table lookup approach.

2.2 Multiple Layers

Missing from most Steiner tree approaches are factors that impact integrated circuit design. First, it should be obvious that most interconnect is not planar, i.e., circuits are fabricated with multiple layers of interconnect metals, normally in alternating directions. A large number of vias are used to perform inter-connections of metals as well as layers, causing relatively high electrical resistance, and consuming a great deal of routing resources.

Fig. 1 presents a routing problem in circuit design. In general, when connecting a set of points, it is typically desirable to utilize minimum length interconnect trees. In the circuit routing, however, some regions may be heavily congested, and it may be worthwhile to trade slightly longer wiring length with lower total cost obtained from less congested region due to lower peak routing demand. The cost of routing in a congested region in the circuit determines the suitable topology of the routing. Geometric techniques can greatly simplify the Steiner problem, but missing to take care of this fundamental issue. Further, each layer may have a unique routing width, giving a distinct cost added to the total routing cost. The Steiner heuristic that forms the basis for our work is adapted to handle differing routing costs on multiple circuit layers [2].

2.3 Congestion in Global Routing

With fixed-die designs, routing congestion has become a significant concern [12]. The routing resources are limited, and if too many connections attempt to use one portion of the routing area, significant detours are unavoidable. In rip-up and reroute based global routers, it’s common to assign resource cost based on utilization [13]. Thus, the “cost” of a connection is not from the physical distance, but the sum of the costs of individual routing tiles (usually determined by Dijkstra’s algorithm [14]). In this paper, we use a simulated annealing algorithm (SA) [28] to select the direction of routing path obtained from the Dijkstra’s algorithm.

The focus of this paper is on quickly finding high quality Steiner trees, under a congestion based cost metric. The problem we address is topology design using global routing based costs, as shown in figure 1(c). In this figure, we face with a routing problem containing three points. While this would be trivial to solve with a simple distance metric, a congested region (with higher cost) may make a spanning tree configuration preferable. There are several effective graph-based Steiner heuristics (such as one by Minoux [15]), but these are designed with no underlying structure. Our research focus on integrated circuit routing, and we present an approach that is as fast as a spanning tree construction, while accurately modeling routing costs.

2.4 Simulated annealing

In this paper, we apply an edge removal heuristic by using a simulated annealing algorithm (SA) [28], where an initial solution is repeatedly improved by making small alterations until further improvements cannot be made by such alterations. Unlike greedy-
type local search algorithms, the SA algorithm can avoid entrapment in a local minimum by allowing occasional uphill moves which deteriorate the objective function value. The uphill move is allowed with the probability given by $\exp\left(-\Delta/T\right)$, where $T$ is a control parameter called the temperature, and $\Delta$ is the difference between objective function values of the current and neighborhood solutions. The temperature is initially set with a certain method and gradually lowered in a predetermined method, called the cooling schedule. The following shows how the SA algorithm is implemented for the global routing problem. The temperature $T$ is initially set high. Therefore, the probability of accepting a move that increases the objective function is initially high. The temperature is gradually decreased as the search progresses. That is, the system is cooled slowly. At the end, the probability of accepting a move that decreases the objective function value becomes vanishingly small. In general, the temperature is lowered in accordance with an annealing schedule. The most commonly used annealing schedule is called exponential cooling, which begins at some initial temperature, $T_0$, and decreases the temperature in steps according to $T_{i+1} = \alpha T_i$, where $0 < \alpha < 1$. Typically, a fixed number of moves must be accepted at each temperature before proceeding to the next. The algorithm terminates either when the temperature reaches some final value, $T_{final}$, or when some other stopping criterion has been met.

3. CONGESTION AWARE STEINER TREE (CAST)

One may anticipate that when considering routing congestion in an interconnect tree, the approach must necessarily rely on shortest path algorithms (and consequently, there would be a dramatic increase in run time). In practice, however, this is not the case.

![Fig.2: The edge removal heuristic.](image)

While working on congestion prediction and estimation, Westra, Bartels, and Groeneveld [16] noted that almost all point-to-point connections generated by a commercial global routing tool contained a single bend. While the optimal path in a graph might be expected to meander, this does not seem to occur frequently. One might expect that this is an artifact of considering via cost as part of the routing objective: if vias are expensive, changes in direction are penalized.

From Fig. 2, given an initial spanning tree, potential connection points between edges and vertices are determined. If a connection between an edge and a vertex is made, it will create a cycle in the tree; the longest (or highest cost) edge on the cycle can then be removed. The global routing problem can be modeled as a grid graph $G(V, E)$, where each vertex $v_i$ represents a rectangular region or module of the chip, so called a global routing cell (G-cell), and an edge $e_{ij}$ represents the boundary between $v_i$ and $v_j$ with a given maximum routing resource $m_{ij}$. A global routing is to find paths that connect the edges, $E$, inside the G-cells through $G(V, E)$ for every net, $(v_i, v_j)$ [19].

![Algorithm CAST L-Shape global routing](image)

Algorithm CAST L-Shape global routing

```algorithm
int layer1, layer2

Cost = Horizontal_routing start and Vertical_routing record
Cost2 = Vertical_routing start and Horizontal_routing record
Minimum Cost = max(Cost1, Cost2)
Routing with Minimum Cost is shown.

// Horizontal_routing
for (layer1 = 0; layer1 <= max_layer; layer1++)
for (layer2 = 0; layer2 <= max_layer; layer2++)
if (cost = cost from $(x_1, y_1)$ to $(x_2, y_1)$) on layer 1
plus cost from $(x_2, y_2)$ to $(x_2, y_3)$ on layer 2
plus cost from $(x_3, y_2)$ to $(x_3, y_2)$ on layer 2
plus cost from layer 2 to $z_2$

// Vertical_routing
if Similar technique from Horizontal_routing is applied for vertical routing (T decision).
```

Fig.3: Algorithm CAST L-Shape global routing.

From Fig. 3, the CAST L-Shape global routing algorithm illustrates global routing in 3-dimension. To understand easily, we demonstrate it in x-y axis, where the MST (Minimum Spanning Tree) has been obtained. With the constraint that point-to-point connections contain at most one bend, we can adapt the edge removal heuristic algorithm to optimize the routing congestion, while maintaining the excellent run times. The edge removal heuristic algorithm of Borah, Owens, and Irwin [3] is chosen. The detail of the algorithm is described in the next section.

3.1 The Edge Removal Heuristic Algorithm

The edge removal (ER) heuristic algorithm that forms the basis of our work is remarkably flexible, as well as easy to implement. The heuristic algorithm begins with a minimum spanning tree, and then attempts to improve itself through a series of passes.

In each pass, we consider introducing a connection between any two vertices with each edge in the graph. This connection would create a Steiner point (the cheapest connection for the vertices and edges),
and also a cycle. Redundant edge is removed from
the graph in order to minimize the connection cost.
The potential connections and gains can be computed
quickly with $O(V)$ using depth-first search algorithm.
After determining possible modifications that can re-
duce tree cost, they are applied in order of benefit; it
is easy to determine if one modification precludes an-
other. Each pass is at most $O(V^2)$, where the depth-
first search algorithm is $O(V + E)$, and the number of
dges, $E$, in the spanning tree is $O(V)$. Typically, the
iterative process fails to find further improvements af-
ter about three passes. An edge-removal operation is
illustrated in Fig. 2.

3.2 Routability and Wiring Length

Routability can be estimated by the number of
over-flows which indicates that routing demand lo-
cally exceeds the available routing capacity [25, 26].
The number of overflow between any two adjacent
units is one.

- Wiring Length is an important metric for place-
ment as well as routing. However, it is less concern for
global routing, as routing all wires with shortest path
algorithms will result in minimum or near-minimum
wiring length [24]. However, there can be huge differ-
ence between solutions of the same wiring length in
terms of routability.

- Runtime is fairly significant, as global routing is sig-
nificantly related to placement and detailed routing.
Global routing is performed first, followed by detailed
routing. Parasitic information from the lower level
design is may be needed to feedback to the higher
level of design flow for the design convergence. With
efficient global routing algorithm, the runtime for the
circuit design can be minimized.

3.3 Adding Congestion to the Heuristic

When considering routing congestion, the obser-
vation that connections are typically L-shaped [16]
becomes very useful. Rather than computing connec-
tion cost using Dijkstra’s algorithm, we can instead
simply sum up the costs of routing tiles of the two
L-shaped paths for a pair of points.

In rip-up and reroute approaches, routing cost is
a function of the demand for graph edges. In many
recent works, cost increases abruptly when demand
reaches (or approaches) its capacity. In the CAST
routing tool, routing cost is a linear function of de-
mand, so that we can pre-compute the costs of series
of tiles. Rather than adding costs in a step-by-step
way, we can compute costs in a constant time. This
method is illustrated in Fig. 4.

Pre-computing routing costs can be time consum-
ing. However, in the case of a global routing, this can
be done once per pass for the router. In [17], Steiner
trees were repeatedly rebuilt, to balance the routing
utilization of multiple layers. Thus, with each pass,
overall routing demand changed slowly. As a result,
we can anticipate only moderate changes in congest-
ion (and routing cost) with each pass of tree con-
struction, and there is little need to rapidly update
the cost surface.

From Fig. 4, congestion map utilization for global
routing is shown in the upper left portion of the fig-
ure. The routing cost is based on resource demand,
rather than the routing distance alone. As most con-
nections each contains a single bend, we can pre-
compute horizontal and vertical costs, in order to de-
termine the cost of any point-to-point connection in
a constant time.

3.4 Global Routing with Congestion Estima-
tion

Our routing approach extends previous work
through the introduction of a congestion estimation
method. This influences the routing of individual
connections, improving the solution quality. In this
section, we first describe the routing method of [11] in
more detail. We then present our method of conges-
tion prediction. The routing costs are influenced by
the congestion estimates; our objective is to disperse
routes from areas that we expect to have routing con-
gestion.

We consider the routing cost to be slightly increas-
ing at the beginning where the demand is still low
until it gets to a certain limit where more routing de-
mand greatly affect to the routing cost; the routing can be obtained with sacrificed cost up to a certain capacity. Beyond this limit, more routing can still be obtained with a different cost rate. In this way, we pay more attention on the routing demand that is getting close to the routing capacity.

Fig. 5 shows the CAST routing cost function. We assign a unit cost to an edge initially and the cost is slightly increase linearly until it reaches 20% of the routing capacity, where the cost starts to rapidly increase linearly until reaching a maximum cost. The parameters \( m_1, m_2, m_3 \) are routing demand scales, \( x \) is number of nets \( n \), \( k_1, k_2, k_3 \) are edge capacities, and \( c_1, c_2, c_3 \) are congestion parameters. \( k_1 \) is the initial step. In this experiment we set \( m_1 = 0.2, c_1 = 0.8, m_2 = 6, c_2 = -22, m_3 = 0.2, \) and \( c_3 = 7. \) If the steps keep repeating on the same route, the cost will increase slowly until it reaches \( k_2 \), where the cost increases dramatically at this stage and beyond. The routing cost behavior in Fig. 6 shows number of repeating routes \( R \) that the cost keeps increasing slowly from number of repeating routes 1-4, then at the 5th repeated route, the cost suddenly increases dramatically to force the process to choose a different route. Then, if the congestion solution is found after a few steps at some other routes, the process can return to this route again while the cost keeps increasing slowly. This method is a modified version of the classical cost function (step function).

As is done in most global routing tools, we monitor the number of routes in our routing graph, and use this to adjust routing cost. We consider this to be an extremely important issue in the construction of an effective global routing tree. In [20], it was observed that having routing cost increase linearly with congestion was far more effective than having a “step” cost function; these observations were confirmed in [27]. Despite this, several recent global routers such as Labyrinth [22] and Albrecht [23] have their routing costs increasing abruptly when the number of routes on a graph edge reaches the edge capacity. Our routing cost function is illustrated in Fig. 6, where the maximum routing cost is set to 10.

The routing costs are influenced by the congestion estimates; our objective is to disperse routes from areas we expect to be congested. Prior to initial routing, the method of Linsker [22] has no indication as to where congestion is to be expected. The first routes may pass through areas that will have high demand, even when there are alternatives. During rip-up and reroute, the routings of some wires impacts the routes considered for others. Our objective with congestion estimation is to minimize the number of poorly routed connections in the early stages of routing, and to provide an effective method when there are multiple routes with similar cost. In [21], a similar technique is considered, but dynamically updated routing cost because no capacity constraints were included with their benchmarks. The intuition for the “linear” cost function is clear: early routes will encounter an uncongested routing graph, and will utilize the resources without consideration for the demands of later routes. Later routes will then detour around congested regions, increasing the overall routing demand considerably. During rip-up and reroute, the connections routed in the early stages will have difficulty in being rerouted, because of the added wire length.

We demonstrate our approach by applying the heuristic algorithm to randomly select nets on ibm01 (ISPD98 IBM Benchmark, grid size = 64×64, vertical capacity = 12, horizontal capacity = 14, and number of net = 11,507 nets) using routing cost based on a global routing congestion map. To evaluate the performance of the CAST, we perform experiments on a set of random nets and compare the routing costs of our CAST, CMST, and ER routing costs. We consider various cases started from 100 to 10,000 nets. These results are shown in Table 1. To our knowledge, there is no other publicly available Steiner tree heuristic algorithm which operates on a global routing graph. Detailed information of the costs is as follows.

- The “ER” or Edge Removal heuristic algorithm [25] cost columns indicate routing congestion cost, relative to a minimum spanning tree constructed with tree length as the objective. The length metric is a traditional comparison; our implementations of the edge-removal heuristic algorithm are comparable to those previously published. Because the Steiner tree is constructed regardless of the routing congestion metric, not all improvements performed by this heuristic algorithm are actually beneficial. As a result, overall solution quality may be degraded.

- The “CMST” or Capacitated Minimum Spanning Tree algorithm [25] cost columns show results when the minimum spanning tree constructed with rout-
other termination criterion has been met. In the solution’s fitness value in a few iterations or algorithm terminates when there is no improvement at initial temperature, $$T_0$$ with the congestion cost. Exponential cooling begins Annealing algorithm, and evaluate the CAST result be shown as follows.

Cost comparisons

Table 1: Cost comparisons

<table>
<thead>
<tr>
<th>n</th>
<th>CAST</th>
<th>ER</th>
<th>CAST−ER</th>
<th>CMST</th>
<th>CAST+CMST</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1255</td>
<td>1272</td>
<td>18.856%</td>
<td>6.830</td>
<td>47.260%</td>
</tr>
<tr>
<td>860</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>900</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>950</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>100</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>105</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>110</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>115</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>120</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>125</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>130</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>135</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>140</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>145</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>150</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>155</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>160</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>165</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>170</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>175</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
<tr>
<td>180</td>
<td>1380</td>
<td>1406</td>
<td>18.335%</td>
<td>6.994</td>
<td>47.257%</td>
</tr>
</tbody>
</table>

Run times for our heuristic algorithm are low. Overall, our approach is $$O(V^2)$$, making it nearly as fast as the best spanning tree algorithms. As a result, our approach is suitable for large circuit designs.

In Table 1, we compare the routing cost of our CAST, ER and CMST approaches. The numbers of nets ($n$) are varying from 100 to 10,000. We compare our CAST algorithm with the capacitated minimum spanning tree (CMST) and the ER algorithm finding that our approach offers impressive reduction in congestion cost in average about 23.1% and 63.4%, respectively. When routing congestion is considered in spanning tree construction, the overall wiring length is slightly higher, but the tree cost is slightly lower. The best result (in terms of interconnect tree cost) are obtained by our Congestion Aware Steiner Trees (CAST). The routing results from our CAST algorithm are graphically illustrated in Fig. 7(a) for the number of nets ranging from 0 up to 1,000, and Fig 7(b) for a high number of nets ranging overall from 0 up to 10,000.

![Fig. 7(a): CAST, ER, and CMST routing costs for small to moderate net size](image)

![Fig. 7(b): CAST, ER, and CMST routing costs for moderate to large net size](image)

In this paper, The CMST technique combines the speed and solution quality of a high quality Steiner heuristic algorithm with the reality of modern routing. Each routing layer has a preferred direction, and
connecting vias have significant cost. We do not discuss obstacles and blockages; which are trivial to add to our formulation (by simply maintaining a count of the number of blocked routing tiles along with the summations). Our method also supports routing costs on a per-layer basis. For routing problems that contain large numbers of obstacles, maze-routing approaches would be more appropriate.

4. CONCLUSIONS

Routing is a key stage for VLSI physical design. This aggressive technology scaling has led to smaller/faster devices, but more resistive interconnects and larger coupling capacitance. Since routing directly determines interconnects (wiring length, routability, congestion, etc). The manufacturability and variability issues such as antenna effect, copper chemical mechanical polishing (CMP) and sub-wavelength printability have also emerged. Again, routing plays a major role in terms of the manufacturing closure. The global routing, as its name implies, is the routing stage that plans the approximate routing path of each net to reduce the complexity of routing task and guide the detailed routing. Thus, it significantly impacts on wiring length, routability and delayed time.

In this paper, we have presented the efficient Congestion Aware Steiner Tree (CAST) heuristic algorithm. From our observation that most global routing connections are simple L-shaped routes, we can use cached routing costs to speed up the calculations. Overall, our method is as fast as the best Steiner algorithms, which performs well when facing with a simple distance metric, and shifts to handle routing congestion costs easily. It also supports preferred direction routing, and the corresponding via costs.

As part of our current work, we are integrating the approach with our global routing tools [18]. We will report results of this integration when the work is completed.

5. ACKNOWLEDGEMENT

This work was supported by Grant for Development of New Faculty Staff Awarded 2001, Rajamangala University of Technology Isarn, Thailand. We also would like to thank Computer Science Dept. at Thomas J. Watson School of Engineering, State University of New York at Binghamton for supporting research facility during the year of 2006-2007.

References
[17] A. Agnihotri, and P.H. Madden, “Congestion...


