A Novel Technique For Sea Of Gates Global Routing

Bharat Krishna*, C.Y. Roger Chen+, Naresh K. Sehgal*
*MS-SC12-606, 2200 Mission College Blvd., Santa Clara, CA 95052
bkrishna@scdt.intel.com, nsehgal@scdt.intel.com
+Dept. of Elec. & Comp. Engineering, Syracuse University, Syracuse, NY 13244
rchen@cat.syr.edu

Abstract

We present a novel global routing and cross-point assignment methodology for sea-of-gates (SOG) designs. Using the proposed congestion driven spanning trees (CDST), and continuously analyzing the congestion at all steps, nets are incrementally globally routed in one of the six well thought of heuristic steps of our methodology. This eliminates the need for rip-up and re-route and enables our approach to achieve good completion rates. We tested our flow on a number of testcases from the industry. The net lengths produced by our flow were compared to the theoretical lower-bound (steiner tree) and were found to be at most 4% worse. We compared the results of creating the net segments by the classical MST verses the CDST and observed an 80% improvement in the number of incomplete nets. Completing the detailed routing using a commercially available detailed router validated the results of the proposed global routing and cross-point assignment.

1. Introduction

The absence of routing channels makes the SOG methodology very suitable for designing very dense high performance integrated circuits. In deep sub-micron VLSI chips, the number of nets has increased significantly and this increases the complexity of the routing problem. Layout routing automation has become very critical to the success of a layout design. It is not only necessary to minimize the layout area of nets, but also to minimize the number of nets not routed by the automation process. Since routing has been proved to be a NP-complete problem [2] , all solutions are heuristic and none guarantees 100% completion. After automated routing of a block, it typically takes a few weeks to manually complete the routing of incomplete nets because of the increased design complexity. Therefore, the primary objective of our heuristic based work is to minimize the number of incomplete nets (and consequently reduce the manual effort), while achieving near-optimal net-lengths.

1.1 Related Work

Many global routing solutions, which use the flow shown in Figure 1(A), have been proposed. In [10] , Lee and Sechen have proposed a 7-step general GR solution that also follows a similar flow. Maze routing is used for re-routing and some choices for routing are made randomly. Their approach focuses on reducing net length and may increase the number of vias and does not account for obstruction. In [4] , Kao and Parng start out by assuming that global routing has constructed the routing trees and assigned GRCs to all nets. Congestion analysis of GRCs is then carried out. A rip up followed by global *re-route* and CPA is iterated for all nets that cannot pass through assigned GRCs because of congestion. Other global routing, such as Steiner tree [3] , SERT [5] , A-Tree [6] , P-Tree [7] , etc, are used for tree construction. These types of solutions determine the tree topology of a single net without consideration of obstruction that may be caused by the presence of other nets. Such solutions are useful for routing of nets that are considered critical and need to be routed before the global routing of all other nets.

Other approaches follow a slightly different approach in that the optimal routing trees are not constructed, but rip-up and re-route is used to relieve congestion. In [12] , Aoki and Murakata propose a methodology that divides nets into from-to segments using the classical minimum spanning tree (MST). The solution assumes that most nets are short which are routed in parallel on many processors. In [11] , Hayashi *et al.* propose a multi-layer routing for the SOG methodology, which solves the layer assignment and GR problem simultaneously. Congestion analysis is carried out only after all net segments are assigned to GRCs and rip-up and *re-route* is used to relieve congestion.

In the approaches where alternative trees are constructed for the ripped-up nets, the initial construction of optimal trees for these nets becomes a wasted effort. New tree construction followed by congestion analysis becomes a loop that is applied until all nets have been globally routed. Rip-up and re-route process induces an element of randomness that is used to eliminate the need to order nets. This is a slow and time consuming method because of the numerous iterations involved with *rip-up* and *re-route*.

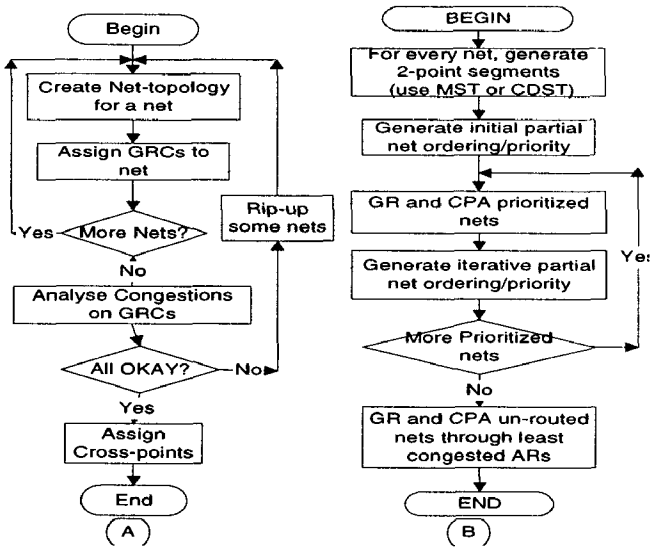


Figure 1: (A) Typical Flow, (B) Proposed GR & CPA flow

2. Our Proposed Approach

We propose a greedy approach to simultaneous global routing and CPA that tackles the problem of net ordering and avoids rip-up and re-route loops. By continuously updating the congestion and the topologies of net segments, a partial net ordering is created. The partial net ordering drives our approach to achieve acceptable results quicker. The proposed approach differentiates from others in the following:

- a new congestion driven minimum spanning tree
- dynamic segment selection for routing consideration without

requirement of any pre-defined net sequence
- focus on routing resources rather than tree construction
- automatic net prioritization to eliminate rip-up and re-route

Our proposed approach to the GR and CPA process is shown in Figure 1(B). By considering the congestion along the routing path of each net segment, we use various heuristics that automatically prioritize nets. Net segments are globally routed and CPA done according to the ordering obtained by these well-defined heuristics. The experiments, which measure the results in the rectilinear metric, i.e. net length, support our approach and show acceptable solutions can be obtained without rip-up and re-route process.

2.1 Problem Definition and Assumptions

The problem as defined in [9] , is to determine for every net which cells are used to route that net (i.e. global route) and to find the exact cross-point on the boundary of each cell that the route uses (i.e. cross-point assignment).

Like many other existing solutions, our solution makes some similar assumptions as follows:

- arbitrary locations of ports & obstructions in the routing area
- two layers for routing with pre-defined routing orientations
- fixed routing area

In addition, we also make the following assumptions:

- minimizing the number of bends is desirable to avoid changing routing layers (reduce the RC delay of a via) and to minimizing metal-migration effect, which can cause problems in high speed circuits.
- creating a dog-leg between two adjacent tracks can be accomplished on the same layer, i.e. not use a via.

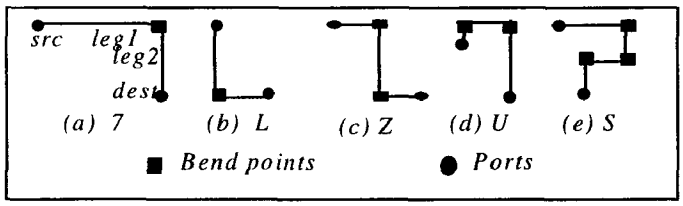


Figure 2: Various ways of connecting two points

2.2 Options for rectilinear connection between two points

Given a net segment between two ports, the net topologies shown in Figure 2 can be used to connect the two ports. A similar set of topologies has been used in other approaches, e.g. [10] , etc. The 7 and L topologies are the two most optimal routes because they have the minimum rectilinear route and the number of bend points is a minimum. The Z and U topologies (Figure 2(c) and (d)) are multi-bend, which can be formed as a combination of L and 7 shapes. This basic idea is extended to connect multiple points by creating pairs of points that are connected together.

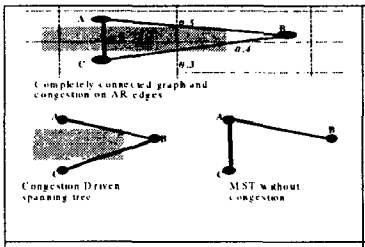


Figure 3: Difference in MST and CDST

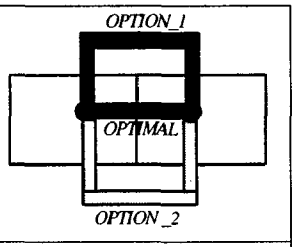


Figure 4 : prioritizing segments in Pass 1

2.3 Proposed Solution for Pairing

An option for segmenting a net is to use the branches of a minimum spanning tree as the two point net-segments. In [8] , Cong *et al.* showed that using the MST is as good as using other more optimal tree algorithms for most global routes in the standard cell designs. In [11] the authors also used the MST for net segmentation. The drawback of using this approach is that a spanning tree is constructed for one net at a time and does not consider the congestion due to other nets. The routes constructed using these segments may be placed in congested regions which are then ripped up and re-routed through alternate regions during congestion relief iteration. We have formulated congestion driven spanning tree (CDST) which considers the possible congestion due to other nets while constructing the spanning tree for a net. In our formulation, the weight of an edge is determined from the following three factors:

- the length of the edge
- congestion due to obstruction
- anticipated congestion due to other net

Since the final routing is rectilinear, the congestion of all edges is also computed from the rectilinear connections between the vertices of the graph. We define the congestion index as the average of C_L and C_7, which are congestion along the L-shaped and the 7-shaped rectilinear path respectively. The congestion due to obstructions, C_e, along a rectilinear path is defined as the maximum congestion along all the GRC edges that the path crosses. Similar to [11] , the congestion on a GRC edge (C_e) as the ratio of obstructed tracks and total number of tracks.

```
PROCEDURE CDST(nets){
  Define and configure GRCs;
  for each GRC{
    Compute hori. & Vert. Congestion;
  }
  for each net{
    compute complete connected graph;
    for each edge in the graph{
      compute congestion index, C;
      assign weight to each edge;
    }
    compute MST;
  }
}
```

Figure 5: Pseudo code for CDST

A track is obstructed if there exists a port or an obstruction in the track. The congestion due to nets is also considered for creating the spanning tree. By modeling a net by its bounding box, the congestion on a GRC due to nets, C_g(GRC) is defined as the total number of net bounding boxes that intersect with the GRC. The congestion along a rectilinear path, C_n, due to nets is then

$$C_n = \max(C_g(\text{GRC})),\ n = L \text{ or } 7$$

The congestion along a rectilinear path, C_r, is then defined as the sum of C_n and C_e , which are both normalized to 100. An example showing the difference between MST and CDST is shown in Figure 3 and the pseudo-code for creating the CDST is shown in Figure 5.

The time complexity for calculating the congestion along the edges is $O(A)$, where A is the number of GRCs. The complexity for calculating the congestion due to the nets is $O(N * A)$, where N is the number of nets. Therefore, the time complexity of the modified minimum spanning tree is $O(E \log E) + O(N * A) + O(A)$. Since E is normally much larger than N or A, the time

complexity of the modified minimum spanning tree is dominated by *O(E log E)*.

```
Pass 1: priority according to a unique "in-
     demand" direction for a port and
     routing all net segments connected to
     this port. Routing done in preferred
     direction only.
Pass 2: priority according to local CPA &
     critical segments discovery. Routing
     these net segments. Routing done in
     preferred direction only.
Pass 3: explore the two optimal topologies
     for all unrouted segments nets. Route
     in the least congested topology. If
     congestion is equal, postpone to a
     later pass.
Pass 4: explore the two optimal topologies
     for all unrouted segments nets. Route
     in the least congested topology. If
     congestion is equal, then pick one.
Pass 5 & 6: For all unrouted segments,
     explore multi-bend route.
```

Figure 6: Proposed GR algorithm steps

3. Complete Algorithm

The proposed solution is composed of a set of six passes and our heuristics define which segments are routed in each pass. For each pass, the proposed solution uses well-defined heuristics to select segments to route. These segments are referred to as *critical segments*. Other segments follow a delayed binding methodology in the succeeding passes and are routed then. The preferred direction of the critical segments is used to determine the direction of the first leg of the route for their topologies.

3.1 Segment Selection in Pass 1

If a net segment has ports within a row or a column, limiting the route to the row or column (see Figure 7) creates the shortest length of this segment. Such segments have a preferred direction (horizontal for the segment shown) and are routed in this step. Ports connected to these segments are classified as critical ports.

Observation 1: Critical ports must be assigned to their desired pins before the non-critical ports, else its routing will be sub-optimal.

3.2 Segment Selection in Pass 2

The second heuristic uses a systematic way of assigning edges to ports and then ordering the ports for each edge. From Figure 7, it is clear that *net x* and *net y* compete for the same cross-point on the right edge. Since *port y* can cross only one edge (right edge), it is prioritized to cross on this edge. This in turn forces *net x* to cross on the top edge. The heuristic in this step uses this information to obtain an optimal edge and pin assignments for all ports in the GRC.

Observation 2: If there is a solution for assigning ports to pins in a GRC, the proposed approach will find that assignment.

3.3 Segment Selection in Pass 3 and 4

In pass 3, congestion is analyzed for all the unconnected segments in both the optimal topologies. If one of the topologies has lessor congestion, then the segment is routed using the topology. In pass 4, for all segments which have a congestion of less than 100% for both of the optimal topologies, the segments is routed using either of the topologies.

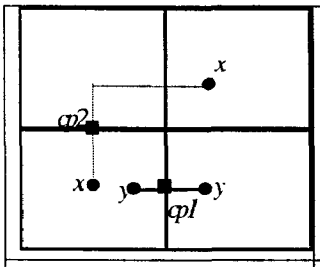


Figure 7: prioritizing segments in Pass 2

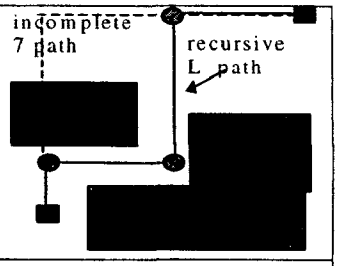


Figure 8: Generating multi-bend Z route in Pass 5

3.4 Segment Selection in Pass 5 and 6

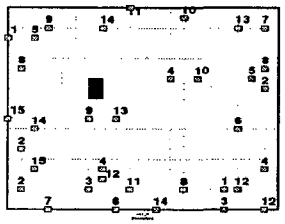
All unrouted segments at this stage cannot be routed using the optimal topologies. The segments are connected by multi-bend routes, which are a combination of 7 and L paths as shown in Figure 8). The start point of the recursive 7 and L path is identified by analyzing the congestion along all the perpendicular segments and the segment with the least congestion is used.

4. Example

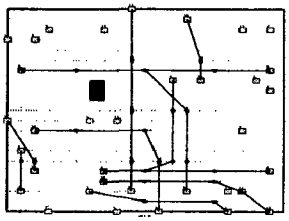
Using the example in [19], we show the steps of our algorithm. We globally routed and assigned cross-points to the example layout shown in Figure 9(a). Step-by-step flow of the proposed heuristics is shown Figure 9(b) to (f). The routing area, which has an area of 16*16, is divided into 4 rows and 4 columns to create 16 GRCs. The example has 15 nets, which are sub-divided into 19 two-point net segments.

5. Experimental Results

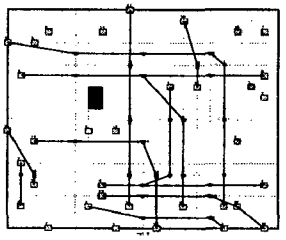
To determine the effectiveness of the proposed CDST, we compared the number of segments routed in each step of the proposed solution for a number of testcases from the industry. From the results shown in Table 1, we observe that the number of segments routed straight (i.e. limited to a row or column) increases and the number of segments routed in U or Z topology (i.e. increased number of bends) decreases with the CDST. Both of this benefit in reducing the number of vias used. Furthermore, the number of nets remaining to be routed manually is also reduced with the CDST reducing the manual routing effort which is generally order of days compared to the seconds taken by the proposed global router.



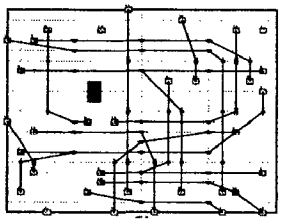
(a) the floor plan



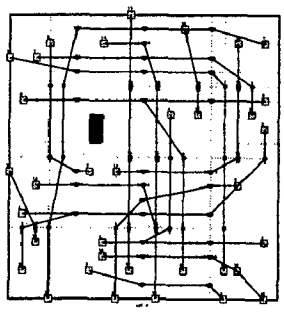
(b) after pass 1



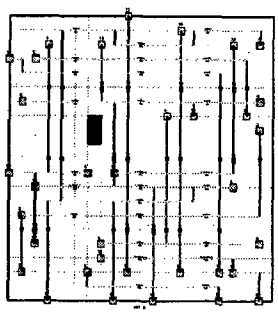
(c) after pass 2



(d) after pass 3 & 4



(e) after pass 5 & 6



(f) Rectilinear Routing

Figure 9 : Step by Step illustration of the algorithm

Table 1 : Net completions rates

Test	Straight		Pass 2		L & 7		Z & U		Manual	
	Cdst	Mst	Cdst	Mst	Cdst	Mst	Cdst	Mst	Cdst	mst
Ra1	126	114	62	69	101	106	0	0	0	0
Jt3	264	250	20	8	11	37	4	4	0	0
Fe1	200	178	333	313	97	136	1	1	9	12
Si1	416	390	127	104	123	172	0	0	0	0
Ix1	661	603	175	179	383	435	0	0	8	10

Table 2 : Net length and runtime comparisions

Test case	# of nets	2-point segments	% comp.		Net length ratio		GR-CPA runtime (sec)	
			cdst	mst	cdst	mst	Cdst	mst
Ra1	315	370	100	100	1.012	1.012	17	8
Jt3	451	606	98.5	98.5	1.005	1.006	218	27
Fe1	622	830	99.9	99.9	1.031	1.03	63	18
Si1	830	1319	100	100	1.033	1.029	717	31
Ix1	803	1470	99.8	99.8	1.012	1.012	275	55
Fs1	1165	1717	98.2	98.2	1.01	1.012	305	51
Fm1	2662	11056	99.9	99.4	1.029	1.04	3815	343

In Table 2, our results measure the rectilinear lengths of all the nets. We compared the lengths of the nets after global routing and cross-point assignment with steiner tree net-length. The results show that the total wire-length is only 4% more than the theoretical minimum in the worst case when the MST is used for segmenting. For the same testcase, CDST segmentation achieves a 25% improvement in the net length comparison and also the number of incomplete nets improves by 80%. Note that the optimal solution is expected to have a total net length more than the minimum.

We used the results produced to generate detailed routing using a commercially available detailed area router. For the ix1 testcase, we created a number of different GRC partitionings by changing the size of the GRCs. For each configuration, we measured the time to run the global router and tested the feasibility of detailed routing using the industrial detailed router. The detailed routing time was also recorded. The results are shown in Table 3. Configuration 7 is performing the detailed routing on the design without any global routing. We can see from the results that using the proposed global routing algorithm can reduce the overall runtime by as much as 50 %. Also we can deduce from the results that making the GRCs very small does not decrease the overall runtime significantly.

6. Conclusion

We have presented a new congestion driven edge-weighting method for generating the spanning trees for use in obtaining global routing solution. The CDST helps reduce the number of vias used in routing. We also defined various heuristics that select net segments in each of the proposed six step heuristic process that efficiently produces near optimal global routing results. By focusing on the routing resources and congestion, rather than routing trees, we obtain the results *without any rip-up and re-route iterations*. Our algorithm greedily minimizes the number of bend points in order to produce high performance global routing and cross-point assignment.

Table 3: Runtimes for various configuration of "ix1"

Config-uartion	# of GRCs	2-point segments for GR	GR-runtime (seconds)	DR-runtime (seconds)
0	200	1470	57	2040
1	400	1572	74	2379
2	800	1533	104	3228
3	800	2019	100	3064
4	1000	2010	121	3173
5	1600	2033	141	3049
6	2000	2114	157	2871
7	1	-	-	4817

7. References

[1] Burstein, M., and Pelavin, R., "Hierarchical Channel Router," Proc. 20th DAC, 1983, pp 591-597.

[2] Garry, M. R., and Johnson, D.S., "The Rectilinear Steiner Tree Problem is NP-complete," SIAM J. Appl. Math., Vol. 32, No. 4, pp 826-834, 1977.

[3] Cohoon, J.P., Richards, D.S., and Salowe, J.S., "A Linear-time Steiner tree routing Algorithm for Terminals on the Boundary of a Rectangle," ICCAD-88, pp 402-405, 1988.

[4] Kao, Wen-Chung, Parng, Tai-Ming, "Cross Point Assignment with Global Rerouting for General-Architecture Designs," IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 14, No. 3, March 1995. pp 337-348

[5] Boese, K.D., Kahng, A.B., Robins, G., "Rectilinear Steiner Trees with Minimum Elmore Delay," Proc. 31st DAC, 1994, pp. 381-386.

[6] Cong, J.J., Leung, K.S., Zhou, D.,"Performance-driven interconnect design based on distributed RC delay Model," Proc. 30th DAC, 1993, pp. 606-611.

[7] Lillis, J., Cheng, C.K., Lin, T. Y., Ho, C.Y., "New Performance Driven Routing Techniques With Explicit Area/Delay Tradeoff and Simultaneous Wire Sizing," Proc. 33rd DAC, 1996, pp. 395-400.

[8] Cong, J., Madden, P., "High Performance Driven Global Routing for Standard Cell Design," Proc. of International Symposium on Physical Design, 1997.

[9] Parng, T.M., Tsay, R.S., "A New Approach to Sea-of-Gates Global Routing," Proc. ICCAD, 1989, pp 52-55.

[10] Lee, K. W., Sechen C., "A new global router for sea-of-gates circuits", Proceedings of the 9th Australian Microelectronics Conference, 1990, pp 233-44.

[11] Hayashi, M., Yamazaki, H., Tsukiyama, S., Nishiguchi N., "A hierarchical multi-layer global router", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E75-A, no.10, Oct. 1992, pp 1294-1300.

[12] Aoki, T., Murakata, M., "A parallel global router for sea-of-gates arrays", 1992 IEEE International Symposium on Circuits and Systems, pp 2997-3000.