Refined Single Trunk Tree: A Rectilinear Steiner Tree Generator For Interconnect Prediction

Hongyu Chen[1] hchen@cs.ucsd.edu
Changge Qiao[2] qiaocg@synopsys.com
Feng Zhou[1] zfeng@cs.ucsd.edu
Chung-Kuan Cheng[1] kuan@cs.ucsd.edu

[1.] Department of Computer Science & Engineering, University of California, San Diego
DCSE, UCSD; 9500 Gilman Dr.; La Jolla, CA 92093-0114
[2.] Synopsys, Inc.
700 East Middlefield Road; Mountain View, CA 94043

ABSTRACT

We devised an efficient and accurate estimation of the rectilinear Steiner minimal tree (SMT), which is an essential building block for on-line and posteriori interconnect prediction. We proposed a new rectilinear Steiner tree generator, Refined Single Trunk Tree (RST-T). Compared with traditional minimal spanning tree based Steiner tree heuristics, RST-T has several advantages. 1. The algorithm runs very fast. Experiments show that RST-T algorithm runs 50 times faster than Prim's minimum spanning tree algorithm for 10-pin nets. 2. The RST-T provides excellent wire length estimation of the optimal solutions. For the nets of no more than 10 pins, the average wire length of RST-T is within 6 percent of the optimal solutions. Actually, for the nets with five or less pins, the wire length of RST-T is optimal. 3. The topology of RST-T remains stable when pin locations deviate. Experiments show that the topologies of routing trees produced by the proposed algorithm is much more stable than the minimum spanning tree and iterative 1-Steiner heuristics.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design aids – *Layout, Placement and routing.*

General Terms: Algorithms

Keywords

VLSI CAD, Routing Estimation, Rectilinear Steiner Tree Algorithm, Refined Single Trunk Tree

1. INTRODUCTION

Interconnect prediction plays an important role in different stages in the design cycle because interconnect delay dominates the path delay in the deep sub-micron era. Particularly, at floorplanning and placement stages design tools need fast yet accurate estimations of the physical parameters (wire length, source to sink distance, etc.) of interconnect nets to guide the tools to optimize the layout. This step is useful because we cannot afford to choose between candidate placements based on a 'real' routing which needs significantly more CPU time than wiring estimation.

Although it is extremely difficult to predict the exact output of various router within very limited computing time, we can still expect that modern routers try to route each net as an SMT with its radius restricted. Hence, estimation of the cost of timing-awareness Steiner tree is an essential part of interconnect prediction in placement and floorplanning stages.

Even though it is widely recognized that minimal spanning tree (MST)-based constructive method produces an excellent estimation of the SMT cost, a lot of previous work still avoid adopting constructive methods under the executing time consideration. A naïve implementation of Prim's algorithm for MST needs *$O(n^2)$* CPU time. By using sophisticated data structure, this complexity can be reduced to *$O(nlogn)$*, but the constant factor is fairly large[13]. Empirical study shows that for the net with less than 100 pins, Prim's algorithm still runs faster than those *$O(nlogn)$* algorithms[18]. To accelerate this time critical task, some previous works use bounding box-based methods to estimate the wirelength [3],[7]. These methods are less accurate, but can achieve linear (for incremental estimation may even be sublinear) time complexity. Another drawback of bounding box-based estimators is that they cannot return an actual topology of the routing tree, which is essential for the accurate interconnect delay estimation. With the wire-size shrinking, the interconnect delay becomes an important factor of the system performance. Many placement and floorplan tools require more accurate estimation on the interconnect delay. Thus, it is important to give a fast constructive method for interconnect estimation. The key part of this constructive method is a Steiner tree generator.

Working as an on-line and posterior interconnect estimation engine, a Steiner tree generator should have the following properties:

- *Fast*: As a part of interconnect prediction subroutine, the Steiner tree generator will be called frequently by placement tools: each time a new candidate placement is generated, the placer needs to evaluate based on its interconnection cost. At least it needs to call the Steiner tree generator to generate the routing trees for those nets whose pin positions are different with previous ones. State-of-the-art VLSI design may contain millions of nets, and the placer needs to search thousands of candidates placements. We expect the Steiner tree generator runs in *O(n)* time or *O(n lgn)* time with fairly small constant, where *n* is the number of pins of nets.

- *Accurate*: To have a good estimation of the wire length and timing, the output of Steiner tree generator for prediction must have good correlation with the output of the router. Since the router can spend much more CPU time on routing tree construction than wiring estimation, it is reasonable to assume that a router will produce a routing tree with its wire length close to SMT and its radius not much longer than the dimension of the net's bounding box. Therefore, the Steiner tree generator used for wiring estimation should produce a Steiner tree with short wire length and small radius.
- *Stable:* Generally, the Steiner tree generator is used to calculate the cost function for other optimizing algorithms (placement or floorplanning). A smooth cost surface can make the optimizing algorithm converge fast or get better solution. Many traditional SMT algorithms start from the MST and then use some heuristic techniques to refine the solution. MST's topology is very sensitive to the pin locations. While some pins' positions change a little, the topology of the MST may change dramatically. For example, in Fig.1(a), we have an MST for a 4 pins net, where pin B is the source, and pins A, D, and C are sinks. We perturb the location of pin D by 0.02. A new MST shown in Fig.1(b) will be resulted, in which the distance from source B to sink D changes from 1.99 to 4.01 while the physical location of pin D only changes 0.02. This phenomenon can damage the smooth property of the cost surface for placer and floorplanner. To avoid this, we want a more stable topology for Steiner tree generator in the context of distance prediction.

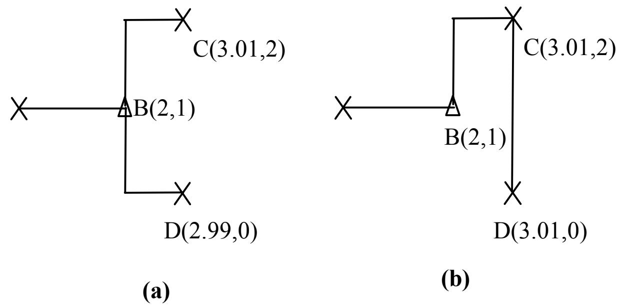


Fig. 1: An example of MST for two 4 pins nets, small perturbation on two pin locations results dramatically change on topology

Therefore, our work aims to find a Steiner tree construction algorithm with the above three properties. In this paper we proposed a simple rectilinear Steiner tree heuristic named Refined Single Trunk Tree (RST-T). Experiments show that our algorithm is at least 30 times faster than the MST algorithms, and gives exact optimal solution for all the nets with no more than 5 pins. For nets with less than 15 pins, our method gives more accurate estimation than MST (within 10% of the optimal cost). Moreover our experiments show that RST-T has much better stability than MST, Kahng-Robins's tree, and SMT.

The rest of the paper is organized in the following way. In Section 2 we give a brief review on the existing rectilinear Steiner routing tree algorithms. In Section 3, we present RST-T as a suitable Steiner tree generator for wiring estimation. Then experimental data are given in Section 4. Finally in Section 5 we give our conclusions and future work directions.

2. EXISTING STEINER TREE ALGORITHMS

SMT problem has been extensively studied in past decades. In 1966, [8] showed that the searching space of the Steiner points could be restricted on Hanan grid points (The grid built by drawing horizontal and vertical lines through each pin). Later Garey etc. proved that this problem is NP-hard [6]. This NP-hardness implies there is no existing polynomial time algorithm that is guaranteed to find the optimal SMT. Currently, the best algorithm takes about one day to derive optimal SMT of 60 or less pins on a single-CPU workstation.

The early work of Hwang [10] showed that wire length of rectilinear minimal spanning tree(MST) is at most 1.5 times of SMT. This result motivated lots of SMT heuristics using MST as a starting point: In [9], Hasan etc. replace neighborhood structures of an independent set of rectilinear MST points by their optimal RST's. Sarrafzadeh and Wong[15] developed a recursive method that keep partitioning the MST into subtrees until the subtree is small enough. M. Borah[8] proposed an edge-based heuristic which repeatedly connects a node to the nearest point on the rectangular layout of an edge. All these MST based algorithms need to construct an MST first, this preprocessing takes $O(n^2)$ or *O(nlogn)* runtime with a fairly large constant.

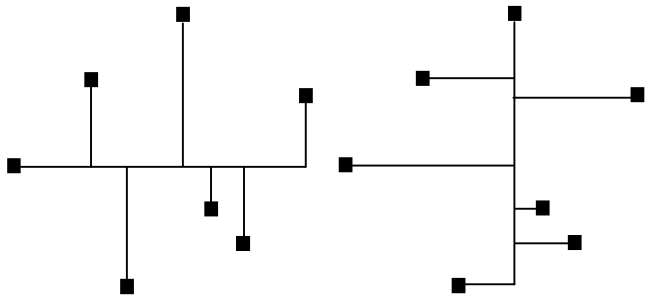
Kahng etc. [12] proposed an iterated 1-Steiner heuristic which makes a significant departure from those MST-based approaches. This algorithm iteratively finds optimal Steiner points to be added to the layout. The time complexity of this algorithm is $O(kn^2)$, where *k* is the number of Steiner points.

Cong etc. [5] proposed a provably good algorithm which can bound the path length by $(1+\epsilon)$ times the radius. And the total wire length is at most $2(1+(2/\epsilon))$ times the SMT. Alpert etc. [1] give a Prim-Dijkstra style algorithm, which produces a hybrid of minimal spanning tree and shortest path tree. These algorithms trade off the wire length and the radius (the longest source to sink distance) to optimize the timing property of interconnect.

3. REFINED SINGLE TRUNK TREE

The proposed RST-T improves from a Steiner heuristic called Single Trunk Steiner Tree (STST) (Fig. 2) [16]. The way to construct STST is straightforward: just connect each pin to a trunk that goes horizontally or vertically through the median position of pins. We can try both of two possible trunk directions and pick the better one as the result. Because of its linear computing time STST has been used in wiring estimation for many years [4][17].

Although the ideas of STST seems primitive, STST model has several advantages as a routing tree generator for wiring estimation: 1) easy to construct; 2) source to sink distance is at most the sum of the length of three edges of the net's bounding box; 3) the topology is stable; 4) for nets with less than 5 pins, its wire length is close to SMT.



(a) STST with a horizontal trunk (b) STST with a vertical trunk

Fig2. Single Trunk Steiner Trees

At the same time, the drawback of STST is also obvious: given a fixed bounding box of net, the total wire length of STST grows at the rate of *O(n)* while for MST and SMT the wire length growing rate is $O(\sqrt{n})$, where *n* is the number of pins in a net. Thus, for the net with large number of pins, the STST gives a routing tree with significantly longer wire length than SMT.

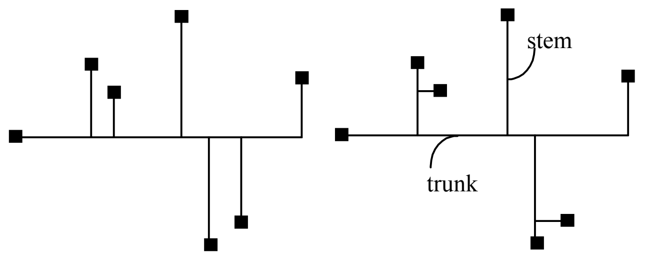


Fig3. Refined Single Trunk Tree and Single Trunk Steiner

To fix this problem, we propose a refining procedure to shorten the wire length of STST. We call the edge that connects the pin to the trunk a *stem* (Fig. 3(b)). Each time we want to connect a pin to the tree, we first check if it is shorter to connect it to the nearest stem than to the trunk. If so, we connect it to the nearest stem instead of connecting this pin to the trunk. The tree produced by this refining procedure is called RST-T. Fig. 3 is an example of RST-T and STST built for the same set of pins.

Fig. 4 is the formal description of the algorithm. After sorting all the pins according to their coordinates, for each pin we can find the neighboring stem within constant time. Hence, the computing time of RST-T is dominated by sorting operating, which can be done in *O(nlgn)* with fairly small constant factor.

We derive following lemmas on the geometrical properties of RST-T:

Lemma 1: Assuming all the pins are randomly distributed in a unit square with uniform distribution, the expected wire length for RST-T is $O(\sqrt{n})$, where, *n* is the number of pins.

When n approaches infinity, the expected distance between a point to its neighboring stem is of order $n^{-0.5}$. Thus, the average cost to connect one point to the tree is $O(n^{-0.5})$. Since the length of

```
Algorithm RST-T
Input: A set of points $P = \{(x_i, y_i)\}$
Output: A rectilinear Steiner tree with all points in P connected

A.  Build an RST-T with horizontal trunk
    1. Set $y_{mid}$ = median of all $y_i$
    2. Set $x_{min} = Min\{x_i|(x_i,y_{mid})\in P\}$, $x_{max} = Max\{x_i|(x_i,y_{mid})\in P\}$
    3. Construct a horizontal trunk from $(x_{min},y_{mid})$ to $(x_{max}, y_{mid})$
    4. Set $U = \{(x,y)|(x,y)\in P \text{ and } y > y_{mid}\}$
           $L = \{(x,y)|(x,y)\in P \text{ and } y < y_{mid}\}$
    5. Sort all points in $U$ according to their $x$ coordinate by
       descending order
    6. Set $P_{ini}=(x,y)$,where $(x,y)\in U$ and (x,y) minimize $|x-x_{min}|+|x- x_{max}|$
    7. Connect $P_{ini}$ to the trunk, going vertical direction first
    8. For all the points in $U$ and to the left of $P_{ini}$, from right to left,
       process point p one by one:
           Connect p to the neighboring stem or to the trunk depending on
           which way is shorter, when we connecting a point to trunk or
           stem, we always go vertical direction first.
    9. For all the points in $U$ and to the right of $P_{ini}$, from left to right,
       process point p one by one:
           Connect p to the neighboring stem or to the trunk depending on
           which way is shorter, when we connecting a point to trunk or
           stem, we always go vertical direction first.
    10. Repeat the procedure from step 5 to step 9, process points in $L$
B.  Build an RST-T with vertical trunk in the similar way to A.
C.  Return one tree with shorter total wirelength from two trees built
    by step A and step B.
```

Fig. 4 RST-T algorithm

the trunk is at most one, the total wirelength of connecting all of the *n* points is $O(\sqrt{n})$.

Lemma 2: For a net with no more than 4 pins, RST-T is an SMT.

For the net with 3 pins, the proposed RST-T is an SMT because the SMT may have at most one Steiner point, and RST-T can find the best one. According to [11], the SMT with 4 terminals may have up to 2 Steiner points. We prove the lemma by showing that RST-T can also find them. Fig. 5 is an example of two possible optimal SMT topologies on four points.

In addition to Lemma2, according to our experiment on hundreds

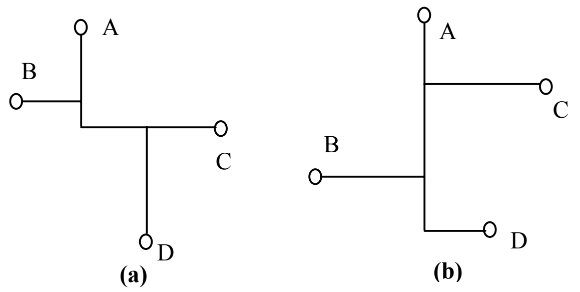


Fig.5. The two possible Steiner tree topologies on four points

of randomly selected test cases, RST-T is always the optimal SMT for 5-pin net. Fig. 6 is examples of different topologies of RST-T for 5-pins nets.

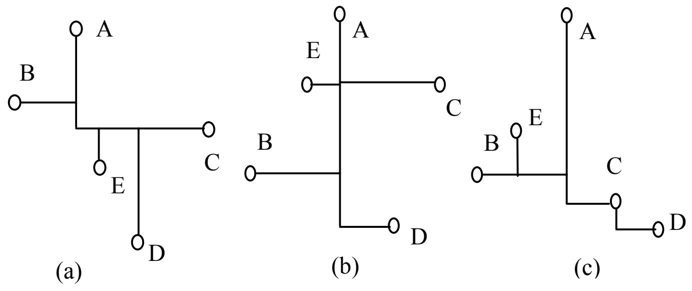


Fig.6. Three possible RST-T topologies on five points

Lemma 3: Assume the net has a bounding box with width w and height h, The path length traversing along RST-T between any two pins is shorter than $max\{3w+2h, 3h+2w\}$.

4. EXPERIMENTAL RESULTS

We implemented the RST-T algorithm using C programming language and compared it with Salowe's SMT algorithm [14], iterative 1-Steiner SMT heuristic algorithm, and Rectilinear MST algorithm. For different n from 3 to 20, 100 different random generated n-pin test cases are solved using all methods. The random test cases are generated from a uniform distribution on a 10000 by 10000 grid. In table 1, the average estimation error and the standard deviations of the errors are showed. In table 2, we compare the running time of different algorithms. In table 3, we compare the stabilities of different methods.

Table 1 shows that RST-T and 1-Steiner heuristic produces the optimal solution for all the 3-pins and 4-pins net. For 5-pins net, RST-T still produces optimal solution and 1-Steiner heuristic's average wire length is about 0.1 percent larger than the optimal one. For net with no more than 10 pins, RST-T's average wire length is within 6 percent of SMT. This error is at least 2 times smaller than that of MST. For all the test cases with less than 16 pins, RST-T gives a more accurate estimation than MST.

In table 2, we compare the running time of different algorithms. Because of the NP-hardness of SMT problem, the exact SMT algorithm runs very slow when the size of net goes large enough. For the largest net we tested, which has 20 pins, RST-T runs faster than exact SMT algorithm with about twenty thousand times speedup. Compared with 1-Steiner heuristc, RST-T can run at least 140 times faster for all test cases. Compared with MST algorithm, which has $O(n^2)$ time complexity, our RST-T achieved at least 10 times of speed up for all test cases. For the net with more than 6 pins, RST-T is more than 30 times faster than MST.

We use a set of 10-pin nets to test the stabilities of different algorithms. For each 10-pin net, we randomly select 3 pins and perturb their locations by 10. We record the changes of source-to-sink distance for each sink. Table 3 shows the average and standard deviation of those changes. RST-T's average source-to-sink distance change is about 6.4% of MST's and 20% of 1-Steiner's. The standard deviation of RST-T is about 50% to 25% of other methods. This experiment implies that RST-T produces a much more stable topology than other three algorithms.

Table 1: Wire length produced by different Steiner tree algorithms

#	1-Steiner		RST-T		MST	
of pins	Ave. Err.	Err. Div.	Ave. Err.	Err. Div.	Ave Err.	Err. Div.
3	0	0	0	0	0.0944	0.0789
4	0	0	0	0	0.1002	0.0474
5	0.0014	0.0062	0	0	0.1041	0.0492
6	0.0012	0.0053	0.0106	0.0264	0.1062	0.0430
7	0.0035	0.0067	0.0142	0.0299	0.1124	0.0412
8	0.0050	0.0078	0.0263	0.0367	0.1130	0.0434
9	0.0038	0.0072	0.0472	0.0370	0.1180	0.0378
10	0.0044	0.0077	0.0594	0.0369	0.1250	0.0399
11	0.0041	0.0074	0.0709	0.0371	0.1237	0.0431
12	0.0036	0.0075	0.0813	0.0373	0.1263	0.0316
13	0.0029	0.0072	0.0897	0.0372	0.1203	0.0308
14	0.0033	0.0073	0.0983	0.0383	0.1177	0.0296
15	0.0025	0.0074	0.1050	0.0395	0.1176	0.0293
16	0.0044	0.0070	0.1246	0.0387	0.1206	0.0307
20	0.0050	0.0068	0.1462	0.0404	0.1251	0.0454

Table 2: Run time of different Steiner tree algorithms

Number of pins	SMT (sec)	1-Steiner (sec)	MST (sec)	RSS-T (sec)
3	2.1e-4	1.1e-4	8.0e-6	7.5e-7
4	3.0e-4	1.7e-4	1.0e-5	7.6e-7
5	4.2e-4	2.0e-4	1.7e-5	7.8e-7
6	6.8e-4	2.3e-4	2.2e-5	7.9e-7
7	9.3e-4	2.4e-4	2.8e-5	8.0e-7
8	1.4e-3	2.7e-4	3.1e-5	8.2e-7
9	2.2e-3	3.2e-4	3.7e-5	8.4e-7
10	3.2e-3	3.8e-4	4.4e-5	8.5e-7
11	4.8e-3	4.6e-4	5.1e-5	8.8e-7
12	7.4e-3	5.2e-4	5.8e-5	9.1e-7
13	1.2e-2	5.9e-4	6.3e-5	9.4e-7
14	2.1e-2	6.2e-4	7.1e-5	9.9e-7
15	3.0e-2	7.3e-4	8.1e-5	1.3e-7
16	5.2e-2	8.2e-4	9.3e-5	1.6e-7
20	8.8e-1	9.4e-2	1.6e-4	2.2e-6

Table3: Stability comparison of different routing tree algorithms

Number of pins	SMT (Exact)	1-Steiner	MST	RSST
Mean change.	77	46	139	9
Std. Of change	0.246	0.113	0.217	0.0536

5. CONCLUSIONS AND FUTURE WORK

We have developed a fast Rectilinear Steiner Tree algorithm for on-line and posterior interconnect prediction. The algorithm produces an optimal SMT for all the nets with no more than 5 pins. For the net with less than 10 pins, the wire length of Steiner tree produced by our algorithm is within 6 percent of optimal solution. That means RST-T can give a very accurate estimation of the wire length for typical VLSI interconnect. The time complexity of RST-T algorithm is *O(nlgn)*. Experimental result shows that this algorithm can run 50 times faster than MST algorithm for 10-pin nets. In addition, RST-T is more stable than MST and Kahng-Robins' heuristic in terms of routing tree's sensitivity to pin locations. This property makes it more suitable to work as a Steiner tree generator for interconnect prediction.

Adapting RST-T to interconnect prediction in the presence of obstacles will be an interesting future research problem.

6. REFERENCES

[1] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng, and D. Karger, "Prim-Dijkstra tradeoffs for improved performance-driven routing tree design," *IEEE Trans. on CAD* 14(7), July 1995, pp.890-896.

[2] M. Borah, R. M. Owens, and M. J. Irwin, "An edge-based heuristic for Steiner routing," in *IEEE Trans. on CAD*, vol. 13, no. 12, pp.1563-1568, Dec. 1994

[3] A. E. Caldwell, A. B. Kahng, S. Mantic, I. L. Markov, and A. Zelikovsky, "On Wirelength Estimations for Row-Based Placement", *IEEE Trans. on CAD* 18(9), (1999), pp. 1265-1278.

[4] A. H. Chao, E. M. Nequist, and T. D. Vuong, "Direct Solution of Performance Constraints During Placement," in *Proc. IEEE Custom Integrated Circuits Conference*, 1995 pp27.2.1-27.2.4

[5] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably Good Performance-Driven Global Routing", *IEEE Trans. on CAD* 11(6), June 1992, pp. 739-752.

[6] M. R. Garey, R. L. Graham, and D. S. Johnson, "The Complexity of computing Steiner minimal Trees," in *SIAM Journal on Applied Mathematics* 32:835-859, 1977

[7] T. Hamada, C. – K. Cheng, and P. M. Chau, "A wire length estimation technique utilizing neighborhood density equations," in *Proc. ACM/IEEE Design Automation Conf.*, 1992, pp. 57-61

[8] M. Hanan, "On Steiner's Problem with Rectilinear Distance", in *SIAM J on App Math* 14:255-265, 1966

[9] N. Hasan, G. Vijayan, and C. K. Wong, "A Neighborhood Improvement Algorithm for Rectilinear Steiner Trees," *in Proc. ICCAS1990* pp2869-2872

[10] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," in *SIAM J. Appl. Math*., pp. 104-114, Jan. 1976

[11] F. K. Hwang, D. S. Richards, and P. Winter, "The Steiner Tree Problem," North Holland Press, 1992.

[12] A. B. Kahng and G. Robins, "A New Class of Iterative Steiner Tree Heuristics with Good Performance", *IEEE Trans. on CAD* 11(7), July 1992, pp. 893-902

[13] F. P. Preparata, and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

[14] J. S. Salowe and D. M. Warme, "Thirty-Five-Point Rectilinear Steiner Minimal Trees in a Day", in *Networks: An International Journal*, volume 25, 1995

[15] M. Sarrafzadeh and C. K. Wong, "Hierarchical Steiner tree construction in uniform orientations," in *IEEE Trans. on CAD*, vol. 11, no. 9, pp. 1095-1103, Sept. 1992

[16] J. Soukup, "circuit layout," *Proc. IEEE*, Oct. 1981, pp. 1281-1304

[17] A. Srinivasan, K. Chaudhary, and E. S. Kuh, "RITUAL: An Algorithm for Performance-Driven Placement of Cell-Based Ics," *IEEE Trans. CAS*, vol. CAS-39, pp. 825--840, Nov. 1992

[18] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/RMST