Wirelength and Memory OptimizedRectilinear Steiner Minimum Tree Routing

Latha N.R
Assistant Professor, Dept. of CSE, B.M.S. College of Engineering, Bangalore, India,
lata.bms@rediffmail.com

Dr. G.R. Prasad
Associate Professor,Dept. of Computer science and Engineering,B.M.S. College of Engineering, Bangalore, India, prasad.cse@bmsce.ac.in

***Abstract*—RectilinearSteiner minimal tree (RSMT) construction is a fundamental issue in designing very large scale integrated Integration (VLSI).FLUTE (Fast Look-Up table) based approach presented a fast and accurate RSMT construction for both smaller and higher degree nets. The model reduces the time complexity for RSMT construction for smaller nets, however for larger nets there exists memory overhead. Since flute basedmodel did not consider the memory requirement in constructing RSMT, the proposed work presents a memory optimized RSMT (MORSMT) construction in order to address the memory overhead for larger nets. Experiments are conducted to evaluate the performance of proposed approach over existing model for varied benchmarks in terms of computation time, memory overhead and wire length. The experimentalresults showthat the proposed model is scalable and efficient.**

Keywords—Minimum spanning tree, Rectilinear Steiner tree, VLSI , wirelength estimation.

I. INTRODUCTION

Rectilinear Steiner Minimal Tree (RSMT) is composed of small set of connected pins through Steiner nodes with minimal cumulative edge size in Manhattan distance for a given set of pins. The construction of RSMT is a major issue in Very Large Scale Integration (VLSI) design and finds it application in interconnect design, placement, floor planning, in computing transmission delay, interconnect delay and also in workload computation. It is also adopted in some global routing strategies to build a routing topography of all nets.

The construction of RSMT for VLSI is considered to be a Non-deterministic polynomial problem[1], asa result rectilinear minimum spanning tree (RMST) has been adopted in some earlier designs by exploring space dimensional aspects.The RMST construction requires a fast tree computing strategy and as RMST does not allow Steiner nodes in tree construction the length of RMST is longer than that of RSMT. In [2] showed that RMST is one and half times greater than that of RSMT with less than 50% in terms of accuracy, which is tolerable in earlier design. However the later design requires good wire length accuracy for which the construction of RSMT is required. In [3] presented a wide range of characteristic for RSMT construction. In [4] and [5] presented an optimal strategy for RSMT construction which is said to have least computation time. In [6] presented a near optimal solution for RSMT construction. However they are computationally very heavy and are not suitable for applications specifically for VLSI design.

Many approaches have been presented to reduce time complexity in constructing RSMT. In [7] adopted spanning graph [8] to aid in building the primary set of spanning tree and obtain finest sets for the edge-which are computed iteratively to eliminate longest edge. In [9] presented a greedy batched technique which improved efficiency and reduced the computation time. The Single Trunk Steiner Tree (STST) is built to connect set of pins to individual trunks which traverse vertically or horizontally through set of all pins, but it is not efficient for medium size pins. In [10] presented refined single-trunk tree for degree up to 5 netsand it is optimally accurate for medium degree nets with fair run time complexity.

In [11] and [12] presented Lookup table based fast and accurate optimal solution for RSMT construction namely FLUTE.Here the nets are recursively broken into sub set of nets. The FLUTE is evaluated for low degree nets and it is suitable for VLSI design. FLUTE is also efficient for high degree nets with runtime complexity of $O(n \log n)$. However for higher degree nets the accuracy of RSMT construction is severely affected.This is due to error induced during net breaking technique. To address this in [13] presented a scalable net partitioning technique, where nets are broken into smaller subset of nets and again merged by adding Steiner nodes. It can handle both smaller and larger degree nets with slight reduction in accuracy but it induces a runtime complexity of $O(n \log^2 n)$. In [14] presented a fast lookup table based RSMT construction which brings a good tradeoff between accuracy and the runtime complexity. In [13] and [14] the memory constraint in building a Look up table is not considered inRSMT construction. The future VLSI designconsist of fixed blocks such as IP blocks, macros, and so on and FLUTE is adopted by these researcher [15] [16] and [17]. In such designs minimizing wire-length and reducing memory overhead is most desired. To address these issues, the proposed work presents a Memory optimized RSMT construction that reduces wire-length and computational overhead complexities.

***The contribution of proposed work can be classified as follows*:**

No prior work has considered Memory constraintin designing RSMT construction. This work presents a Memory Optimized RSMT construction.

The proposed model reduces the wire-length and computation time in constructing RSMT.

The proposed model is evaluated considering different benchmarks [14] and proves that the proposed model is efficient considering all benchmark in terms of Memory overhead, Computation time and Wire-length.

The paper organization is as follows: In Section 2 literature survey is carried out. The Section 3describes the work carried out so far. The proposedMemory optimized RSMT construction model are presented in Section 4. The experimentalstudy considering various benchmarks ispresented in penultimate section. The concluding remarks and future work is discussed in the last Section.

II. LITERATURE SURVEY

Recently many approaches have been presented to minimize wire-length to improve routing performance of VLSI which is surveyed below.The model presented in [13] and [14] did not consider memory performance. FLUTE technique is adopted for the fast congestion driven Steiner tree creation to overcome the existing problem with [18]. It shows the improvement based on runtime complexity. [19] and [20] proposed an approach for solving the congestion problem in global routing using game theory method. This method is also applied for reducing the runtime complexity in clustering approach of VLSI routing. Various clustering based tools are studied in [20] and the work presents a scalable approach of clustering for reducing the cycle time, wire length and optimize the system performance. But clustering approach is a time consuming process and to overcome the time constraint [20] presented a heterogeneous computing model and parallel approach of clustering. It is implemented for both CPU and also work well with GPU system. Performance of the system shows that fast execution is achieved through GPU processing as compared to serial execution butmemory constraint is not considered hence induced memory access time.

To overcome the above short coming this work presentsan memory optimized RSMT construction model for VLSI design which is presented in next section.

III. PROPOSED MEMORY OPTIMIZED RSMT MODEL

In this section we present a memory optimized RSMT construction that reduce wire length and computation time. As similar to [14], let us consider that the size of each sub tree is divided based on memory optimized treeand the Memory optimized tree takesMemory and spanning tree as input. Firstly it computes the least overhead edges (using Memory optimized Spanning graph) and select one of the node as its root. The node which is closer to the root node is considered as parent node by adopting child-parent relationship along the edges using depth-first search and divide and conquer approachwhich results in optimization of memory for larger degreenets.The notations and symbols used are as shown in Table 1 and the Memory optimized RSMT model is shown in Fig. 1.

Let us consider a graph $H(N, M)$, where M and N depicts a set of ordered pair of edges and nodes respectively. Let $m = |E|$ and $n = |V|$ represent set of edges and nodes respectively.

First, we construct and initialize a spanning graph H by adding Steiner nodesα and are considered to be connected to all nodes in H. Then divide-conquer approach is adopted to build a Memory optimized tree of graph H.

The memory optimized divide conquer approach takes Memory S, Spanning graph G of H and graph H as an input and obtain a tree Gas output which is a depth first search tree of H, where G is retained in memory and H is kept in disk. The algorithm first computes whether graph Hcan satisfy memory optimization requirement, so that H can be loaded into S and evaluates Memory optimized tree G of H using available-memory optimization strategy and obtain G. Or else if does not obtain any G, the model further computes Memory optimized tree G of H by dividing Memory optimized tree G by using divide and conquer approach.

To obtain an efficient memory optimized tree the legal dividend of H must be computed which is set to false initially as shown in flowchart in Fig. 1. Then the present spanning graph G is optimized with respect to H until G is a Memory optimized tree G of H or we obtain a legal dividend of H on spanning tree G is obtained.

Here the dividend is modeled by invoking dividend optimization in order to obtain a graph division $H_0, H_1, H_2, \dots, H_d$ of H with resultant spanning graph $G_0, G_1, G_2, \dots, G_d$. The dividend optimizer also evaluates a Memory optimized graph μ in merge operation.

The dividend is said to be legal only if $d > 1$ as shown in flow chart in Fig. 1. Once the legal tree division is obtained, the memory optimized tree G_q is computed for all sub-graph H_q using divide and conquer approach in a recursive manner. Then by combining all memory optimized tree $G_q$of H_q based on $\mu$the memory optimized tree G is computed and obtain G as Memory optimized tree of$H$. The overall flow of proposed Memory optimizedRSMT construction is shown in Fig. 1.

TABLE I. NOTATION AND SYMBOL USED

Symbol Used	Abbreviation
H	Graph
N	Set of nodes
M	Set of edges
$H(N\|M)$	It is a directed graph
G	Spanning graph
α	Set of Steiner nodes
S	Memory Available
μ	Memory optimized subtree graph
d	Number of subtree

The performance study of theproposed approach is presented in next section.

IV. SIMULATION RESULTS AND ANALYSIS

The MORSMT algorithm is implemented in C++ object oriented programming language. The GCC compiler is used to compile the code. The eclipse Kepler IDE used for running the algorithm. The system environment used to run the algorithm is Centos 7.0 Linux operating system, 3.2 GHz, Intel I-5 Quad core processor and 16GB RAM. The IBM benchmark [12] is considered for evaluation which is shown in Table 2. The experiment is carried out to evaluate the performance of MORSMT over existing approach [12] in terms of wire-length, computation time and memory overhead.

TABLE II. BENCHMARK DETAILS

Benchmark Circuit Case	Number of Net	maximum degree	Average Degree
IBM01	14111	3.58	42
IBM02	19584	4.15	134
IBM17	189581	4.54	36
IBM18	201920	4.06	66
Average	106299	4.0825	69.5

In Fig. 2 the computation performance of both proposed MORSMT and Existing approach for varied benchmark is evaluated. The outcome shows that MORSMT performs better than the existing approach in terms of computation time reduction for all the cases. An average improvement of 31.71% is achieved by the proposed approach over existing approach.

In Fig. 3 the wire length performance of both proposed MORSMT and Existing approach for varied benchmark is evaluated. The outcome shows that MORSMT performs better than the existing approach in terms of wire-length reduction for all the cases. An average reduction of 0.02% is achieved by the proposed approach over existing approach.

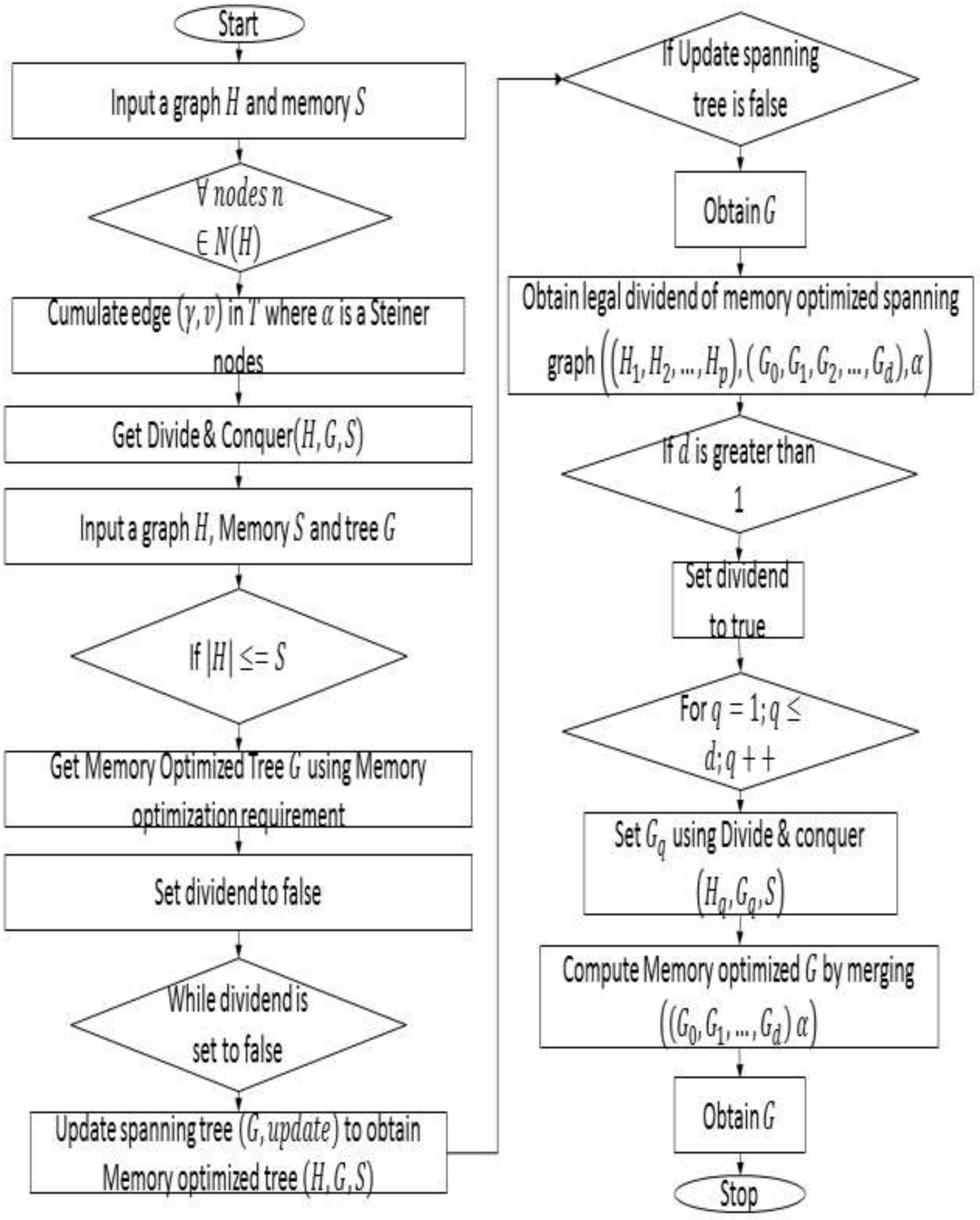
Start
Input a graph H and memory S
$\forall$ nodes $n \in N(H)$
Cumulate edge (γ, v) in T where α is a Steiner nodes
Get Divide & Conquer(H, G, S)
Input a graph H, Memory S and tree G
If $|H| \leq= S$
Get Memory Optimized Tree G using Memory optimization requirement
Set dividend to false
While dividend is set to false
Update spanning tree $(G, update)$ to obtain Memory optimized tree (H, G, S)
If Update spanning tree is false
Obtain G
Obtain legal dividend of memory optimized spanning graph $((H_1, H_2, \ldots, H_p), (G_0, G_1, G_2, \ldots, G_d), \alpha)$
If d is greater than 1
Set dividend to true
For $q = 1; q \leq d; q++$
Set G_q using Divide & conquer (H_q, G_q, S)
Compute Memory optimized G by merging $((G_0, G_1, \ldots, G_d)\, \alpha)$
Obtain G
Stop

Fig. 1: Memory Optimized based RectilinearSteiner Minimum Tree Construction

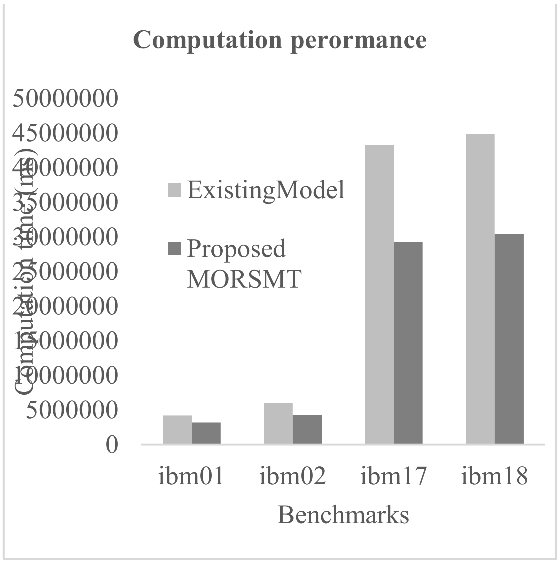

Fig. 2: Computation time performance for varied wire length

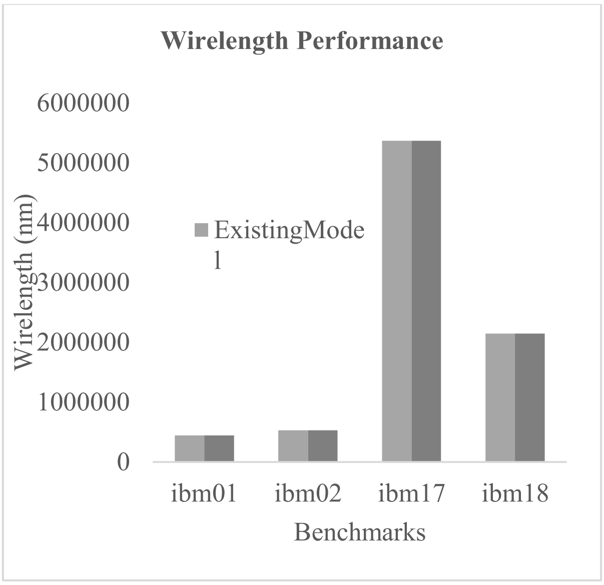

Fig. 3: Wire-length performance for varied wire length

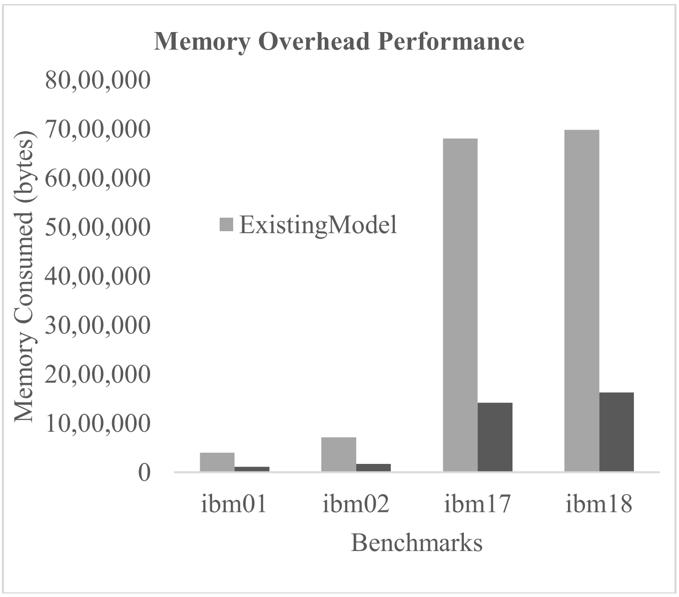

Fig. 4: Memory overhead performance for varied wire length

In Fig. 4 the Memory overhead performance of both proposed MORSMT and Existing approach for varied benchmark is evaluated. The outcome shows that MORSMT performs better than existing approach in terms of memory consumption for all the cases. An average reduction of 77.71% is achieved by the proposed approach over existing approach.

CONCLUSION

This work presented a Memory efficient RSMT construction.The proposed model is an improvement of original FLUTE. The Memory optimization is not considered in RSMT construction of FLUTE and it adopts breadth first search to find minimum spanning tree as a result it induces memory overhead. To address this issue the proposed work adopts divide and conquer and depth first search to find the minimum spanning tree. The computation time is reduced due to less I/O disk access. The model also supports computation for larger degree net with limited memory utilization. Theexperiments are conducted to evaluate the performance of proposed approach over existing approach for varied benchmarks. The outcome shows significance performance improvement of 0.02%, 77.71% and31.71% over existing approach in terms of wire-length, memory overhead and computation time reduction respectively. The future work would consider evaluation for different benchmark and also present memory optimization for RSMT construction on multi-core environment such as CPU/GPU in order to further reduce computation time.

REFERENCES

[1] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide tothe Theory of NP-Completeness. New York: Freeman, 1979.

[2] F. K. Hwang, “On Steiner minimal trees with rectilinear distance,” SIAM J. Appl. Math., vol. 30, no. 1, pp. 104–114, Jan. 1976.

[3] F. K. Hwang, D. S. Richards, and P. Winter, “The Steiner tree problem,” in Annals of Discrete Mathematics. Amsterdam, The Netherlands: Elsevier, 1992.

[4] D.M.Warme, P.Winter, and M. Zachariasen, “Exact algorithms for plane Steiner tree problems: A computational study,” in Advances in Steiner Trees, D. Z. Du, J. M. Smith, and J. H. Rubinstein, Eds. Norwell, MA: Kluwer, 2000, pp. 81–116.

[5] GeoSteiner—Software for ComputingSteiner Trees. [Online]. Available: http://www.diku.dk/geosteiner

[6] I. I. Mandoiu, V. V. Vazirani, and J. L. Ganley, “A new heuristic for rectilinear Steiner trees,” in Proc. IEEE/ACM Int. Conf. Comput.-Aided Des., 1999, pp. 157–162.

[7] H. Zhou, “Efficient Steiner tree construction based on spanning graphs,”in Proc. Int. Symp. Phys. Des., 2003, pp. 152–157.

[8] H. Zhou, N. Shenoy, and W. Nicholls, “Efficient spanning tree construction without Delaunay triangulation,” Inf. Process. Lett., vol. 81, no. 5, pp. 271–276, 2002.

[9] A. Kahng, I. Mandoiu, and A. Zelikovsky, “Highly scalable algorithms for rectilinear and octilinear Steiner trees,” in Proc. Asian South Pacific Des. Autom. Conf., 2003, pp. 827–833.

[10] H. Chen, C. Qiao, F. Zhou, and C.-K. Cheng, “Refined single trunk tree:A rectilinear Steiner tree generator for interconnect prediction,” in Proc. ACM Int. Workshop Syst. Level Interconnect Prediction, 2002, pp. 85–89.

[11] Chris Chu. FLUTE: Fast lookup table based wirelength estimationtechnique. In Proc. IEEE/ACM Intl. Conf. on Computer-Aided Design, pages 696–701, 2004.

[12] Chris Chu and Yiu-Chung Wong. Fast and accurate rectilinear Steiner minimal tree algorithm for VLSI design. In Proc. Intl. Symp. on Physical Design, pages 28–35, 2005.

[13] Wong, Yiu-Chung, and Chris Chu. "A scalable and accuraterectilinear Steiner minimal tree algorithm." VLSI Design, Automation and Test, 2008. VLSI-DAT 2008. IEEE International Symposium on. IEEE, 2008.

[14] Chu, Chris, and Yiu-Chung Wong. "FLUTE: Fast lookup tablebased rectilinear steiner minimal tree algorithm for VLSI design."Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 27.1 (2008): 70-83.

[15] Hao Zhanga, Dong-yi Yea, Wen-zhong Guo ”A heuristic for constructing a rectilinear Steiner tree by reusing routing resources over obstacles” Volume 55, Pages 162–175, 2016.

[16] P. P. Saha, S. Saha and T. Samanta, "An efficient intersection avoiding rectilinear routing technique in VLSI," 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, 2013, pp. 559-562.

[17] G. Ajwani, C. Chu and W. K. Mak, "FOARS: FLUTE Based Obstacle-Avoiding Rectilinear Steiner Tree Construction," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 2, pp. 194-204, Feb. 2011.

[18] M. Pan, Y. Xu, Y. Zhang, and C. Chu, "FastRoute: An efficient and high-quality global router," VLSI Design, vol. 2012, 608362, 2012.

[19] Umair F. Siddiqi, Sadiq M. Sait, and Yoichi Shiraishi, "A Game Theory-Based Heuristic for the Two-Dimensional VLSI Global Routing Problem," Journal Of Circuits Systems And Computers, vol. 24, no. 6, 2015.

[20] Umair F. Siddiqi, and Sadiq M. Sait, "A Game Theory Based Post-Processing Method to Enhance VLSI Global Routers," IEEE Access, vol. 5, pp. 1328–1339, 2017.

[21] A. Momeni, P. Mistry and D. Kaeli, "A parallel clustering algorithm for placement," Fifteenth International Symposium on Quality Electronic Design, Santa Clara, CA, 2014, pp. 349-356.