Optimal memory utilization and wire length minimization for reducing energy consumption of VLSI circuits.

Mohamed Azeem Hafeez
Research Scholar PACE-Mangalore

Dr. Aziz Musthafa B
Department of Computer Science and Engineering
BIT-Mangalore

***Abstract*—the main component of computation model are energy, space, and time. Minimizing them is challenging in building efficient processing design. This work overcomes the problem of using caching utilizing reversible logic circuits (RLC). Further, high communication overhead among cache and main memory affects performance and incurs high heat in VLSI component such as GPU, CPU etc. Thus, limiting or efficient usage of memory resource is most desired. For using RLC, it is important to reduce the delay and wire length of VLSI circuit. This work present an Information processing unit (IPU) using logically reversible circuit using adiabatic reversible Toffoli gate. For minimizing path delay due to presence of faults in circuit design Elmore model is used both at switch level and at wire level and linear model is used at the gates. Further, for bringing tradeoff between memory minimization and delay (processing time) requirement a delay aware rectilinear Steiner minimum tree (RSMT) is created. Experiment are conducted on industrial circuits in the ISPD98 benchmark data. The outcome attained show the proposed model reduces wire length, processing time and memory usage when compared with existing model. Thus, our model aid in improving thread-level and instruction-level parallelization under shared memory environment such as GPU.**

***Keywords*—Cache memory, Primary memory, Rectilinear Steiner tree, Reversible logic circuit, Slack and Slew constraint, VLSI .**

I. INTRODUCTION

In standard PC design where a broadly used Information Processing Unit is utilized, the information kept in the memory is preloaded to a cache memory (CM) placed or kept on the chip with the Information Processing Unit. The purpose behind the information preloading is the contrast among the speed of calculation of the Information Processing Unit and the speed of perusing information from the primary memory (PM). Though the Information Processing Unit works on the maximal probable clock frequencies, the information recovery from the memory needs a session instance bigger than one clock-cycle. Subsequently, so as to amplify the Information Processing Unit execution the right information must be exchanged from the PM to the CM with more old information being expelled/eliminated from the CM.

A few heuristic CM optimization method is being presented with the objective of anticipating which information ought to be erased from the CM (and which information ought to be removed) when the Information Processing Unit demands new information [1], [2]. The stacking of the new information to the CM prerequisite deleting information that are presently not being utilized by the Information Processing Unit but rather that may be needed over in later phases of the calculation. Along these lines similar information can be exchanged from the PM to the CM on various occasions amid Information Processing Unit activity. Such repeated request to the PM make a bottleneck (blockage) that restricts the general speed at which the Information Processing Unit can process the information and produces heat that could be conceivably impeding to the remainder of the IC.

The high use of the CM prompts both an extremely high heat consumption along with to a pattern of continuously expanding the CM size; Since the CM is build using flip-fops its expense is ending up exceptionally high. All things considered, CM isn't just an overhead on the general execution of the Information Processing Unit yet in addition with expanding traffic the head produced can in the end lead to major issues. Along these lines the structure of a CM optimized logic Information Processing Unit could enhance the heat dissemination and in this manner permit incorporating more components into a similar region.

The utilization of reversible logic (RL) is non-existent in current advances since 1) the innovation needed for constructing/structuring ideal reversible circuits (RC) isn't always compatible or accessible with modern circuit or technologies, 2) the need for intelligent (logical) and control (power) reversibility comes at too high a cost (transistor overhead, high wire size, and generally low power saving corresponding to execution requirement).

In [3], came with reversible logic circuit design for overcoming cost and memory overhead in using cache memory. However, these model cannot be used for modern large VLSI circuit [4], [5]. Along with, must resolve routing challenges such as power, propagation delay, buffer insertion and timing constraint. Thus, accuracy of RSMT construction is severely affected for higher degree nets. Further, fault or error exist in modern circuit design [6]. In [7] came up efficient net breaking technique to overcome the error induced during net breaking process. Similarly, [8] presented a fast lookup table model to bring good tradeoffs among computational complexity and accuracy. Both [7] and [8] for constructing Look up table did not considered memory constraint problem. The modern VLSI architecture comes with fixed blocks such as macros, IP blocks etc. and FLUTE is used by various methods [9] [10] and [11]. Considering these architecture reducing memory overhead and minimizing wire length is most preferred [12], [13], and [14]. Further, the modern circuit comes with presence of obstacle. Thus, it is even more challenging to minimize wire length on such environment. Thus, placing buffer closer to obstacle is a probable solution as buffers cannot be placed over the obstacle. However, this will lead to logical and transistor delay [15]. Further, obstacle avoidance RSMT model has been presented in recent times [16], [17], [18], [19], and [20]. However, these model induces overhead, as it consider IP blocks as routing obstacles. Along with, induces delay in routing path. As a result, routing over the block is an effective solution provided it meets design should be aware of signal integrity for longer wire length. Thus, it is important to construct RSMT without violating slew constraint. In [21], showed addressing time constraint can solve slew constrain problem [22]. However the slew constraint depends on delay and wire length size. Thus, the traditional obstacle avoidance RSMT cannot be used for obtaining global routing strategy. Since it doesn't consider the delay perquisite. Along with routing over the blocks cannot guarantee slew violation and interconnect timing perquisite. The strategy to overcome these issues is to minimize the resource required and minimizing communication overhead.

For overcoming research issue, this manuscript present a delay aware RSMT construction that overcomes slack constraint and minimize routing resource (gate). The memory and delay aware RSMT (MDA-RSMT) for obtaining delay aware path is done by minimizing amount of routing resources (buffers) and meeting slack constraint, and memory overhead is addressed by using method in presented in [7], [8]. Our model can overcome the global routing problem [23] with minimal gates. Further, incorporates a slew calculation method presented in [24] into our approach. The proposed memory and delay aware RSMT (MDA-RSMT) minimize delay, routing resource and wire length. Along with, improves memory utilization and reduces computation time.

The manuscript is articulated as described: The proposed memory and delay aware RSMT models are presented in Section two. In section III the result and discussion is presented. In last section conclusion and future work is presented.

II. PROPOSED MEMORY AND DELAY AWARE RSMT MODEL

This section present memory and delay aware RSMT for identifying delay aware path with minimal computation and resources for RLC [6]. Firstly, this work generate a delay aware topology for obtaining delay where path and influx time and each nodes. Then, using this information a rectilinear Steiner tree is created. Further, our approach consider routing over the block as depicted in Figure 2 instead of using routing around the obstacle as depicted in Figure 1. Then, to improve slack and slew constraint outcome (i.e., minimize error) delay optimization is carried out on reconstructed topology. Then, routing is carried out with less delay path and cost.

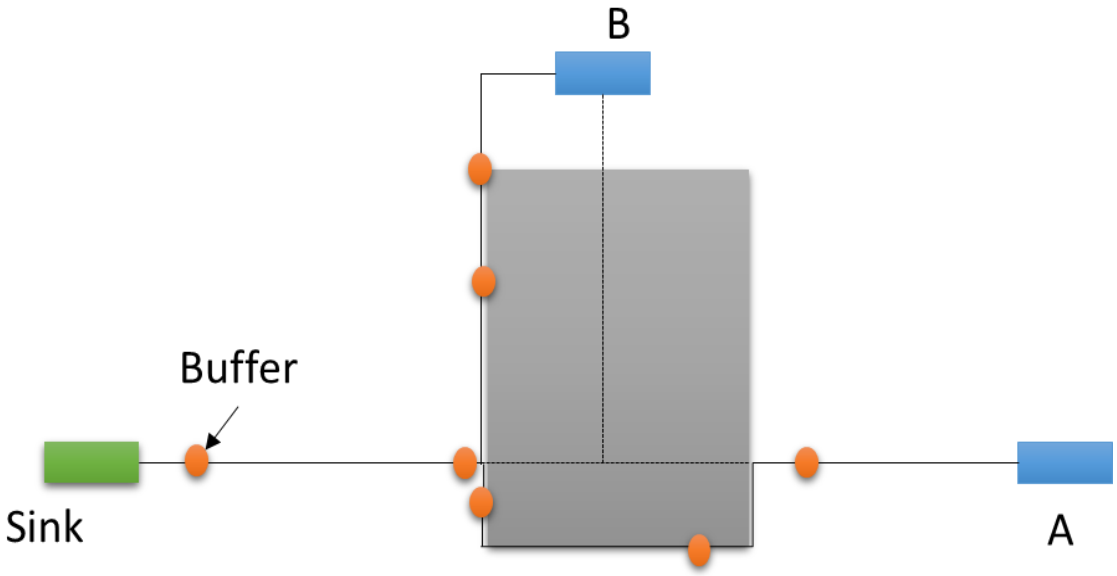

Fig. 1: Existing obstacle avoiding RSMT with two sink A and B and one root sink.

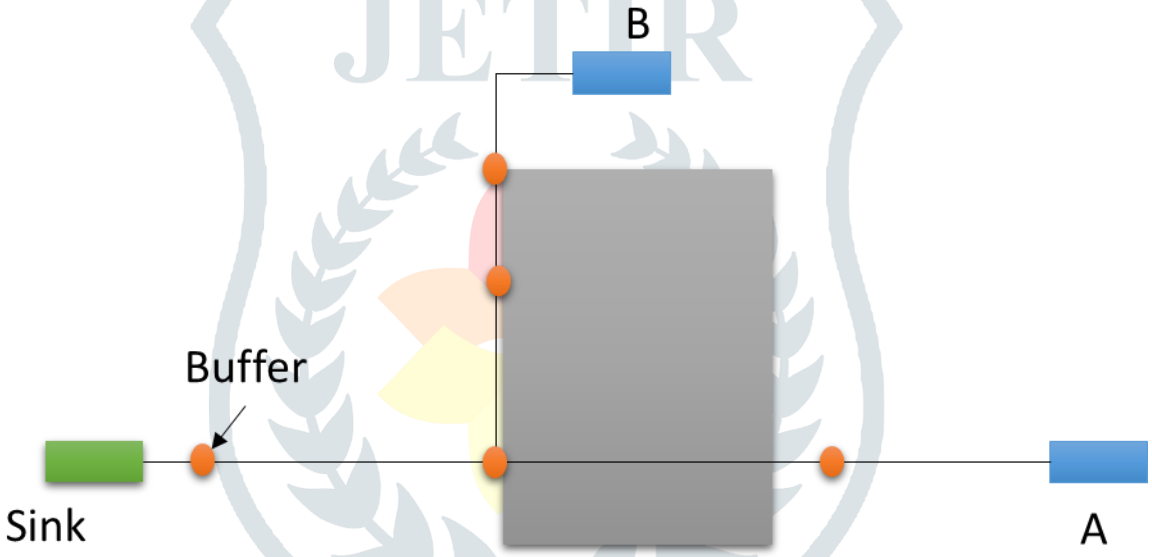

Figure 2: Proposed over the block Memory and delay aware RSMT with two sink A and B and one root sink.

Let assume a communicating/transmitting area with $m + 1$ pins which composed of set of pins of net $K = \{x_0, x_1, \dots, x_m\}$, where x_0 depicts the distinct root nodes and rest of the nodes are the sink nodes. Further, consider a set of non-intersecting rectilinear obstacle in a communicating area U. Along with, assumes entire $x_q \in M$, x_q doesn't exist in the region influenced by K. The obstacle K is used generally for largely dense and complicated cells.

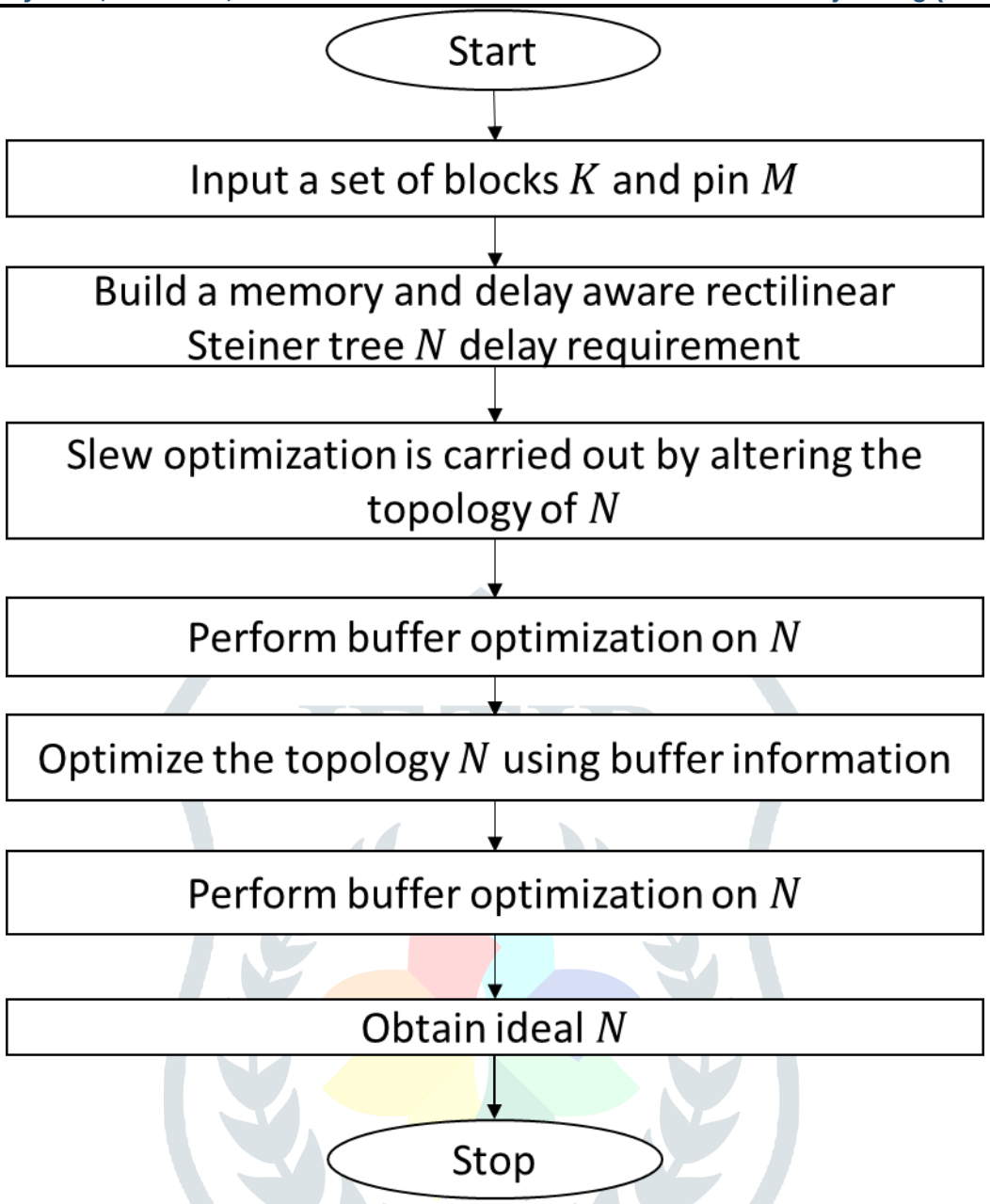

Figure 3: Memory and delay aware RSMT for routing over the block model

The flow diagram of routing over the block using MDA-RSMT is depicted in Figure 3. Using flow diagram presented in figure 3, the proposed approach produces a memory and delay aware graph $N(I,J)$ to join all pins in M. Here J depicts set of horizontal edges and vertical, and I depicts set of sink nodes. The graph set $G = \{N_1, N_2N_3, \dots, N_k\}$ is assumed to be presented within obstacle provided N intersect with obstacle K. The graph G is depicts as inline graph. Similarly, the outline obstacle graph N is depicted as $\mathbb{G}$. The memory and delay aware graph $N_p(J_p, I_p)$ is evaluated utilizing N post insertion of nodes $\mathbb{J}$ that are associated for buffer chosen from delay aware path table P and $J_p = J \cup \mathbb{J}$. Further, there exist buffers along communication path and distinct path $O(x_0, x_q)$ from x_0 to x_q in Steiner graph which can segmented further into sub-path graph. Each segmentation process is composed of set of Steiner nodes, edges associating Steiner nodes, a set of slack aware (optimization) sinks, and slack aware sinks.

The total/overall delay among communicating route can be established by adding the latency or time induced at each level. For measuring time induced at each level and delay for wires Elmore model is used. Further, this work uses switch level delay computing method for computing delay induced for adding gates or buffer. Thus, the overall delay induces at each level of communicating path is computed using following equation

$$n(r(w), s) = \sum_{i=(q,t)\in o(i(w),w)} u_i k_i \left({}^1\!/_2\, v_i k_i + K_w(t)\right) + U_p V_r\big(r(w)\big) + R_p \quad (1)$$

The overall communication delay can be established by adding time induced at each sub-path of communicating route which is estimated using following equation

$$r(x_0, x_q) = \sum_{w \in \mathbb{J} \cap o(x_0, x_q)} n(r(w), w) \tag{2}$$

The worst negative slack $\mathbb{Z}$ is estimated using following equation

$$\mathbb{Z}(N) = min\{0, Z\} \tag{3}$$

where Z is the worst slack parameter condition which is estimated using following equation

$$Z = min\{\mathcal{S}(x_q)\} \; where \; 1 \leq q \leq m \tag{4}$$

where $slack(x_q)$ is the slack of node a_y which is computed as follows

$$\mathcal{S}(x_q) = D(x_q) - r(x_0, x_q) \tag{5}$$

where D is the buffer arrival time.

The slew rate is evaluated utilizing method presented similar to [24] which is done using following equation

$$J_{v_{out_0}} \left(\sqrt{J_{volt_{out}}{}^2 + + J_{step}(volt_{in}, volt_{out})^2} \right) \tag{6}$$

where $J_{volt_{out_0}}$ is the slew at any node $volt_{out_0}$, $J_{volt_{out}}$ is the output slew at node $volt_{out}$ and $J_{step}(v_{in}, v_{out})$ is the step slew from $volt_{in}$ to $volt_{out}$. The step slew is estimated using following equation

$$J_{step}(volt_{in}, volt_{out}) = ln9 * Elmore(volt_{in}, volt_{out}) \tag{7}$$

Further, experiment are conducted to evaluate proposed MDA-RSMT model over standard model. The outcome shows MDA-RSMT uses less memory, less number of gates, reduces wire length and computation time when compared with existing model. Thus, the MDA-RSMT is efficient which experimentally proven in below section.

III. SIMULATION RESULT AND ANNALYSIS

This section present performance evaluation of MDA-RSMT over standard routing method. The MDA-RSMT model is developed using C++ programing language and experiment are conducted on Linux platform using Centos 8.0 and above, with Intel Pentium class processor and 8 GB RAM. Firstly, the experiment are conducted using benchmark from [25] and result are compared with [25] in terms of wire length considering fixed obstacle/blocks and different block size. The benchmark used for case 1 with fixed block size is described in Table I. The pin size is varied from 10 to 100 considering fixed block size of 10 and experiment are conducted. The outcome attained shows the proposed MDA-RSMT reduces wire length of 14.35% when compared with standard model [25] which is shown in Table II. The benchmark used for case 2 with different block size is described in Table III. The pin size is varied from 10 to 200, block size is varied from 500 to 100 and experiment are conducted. The outcome attained shows the proposed MDA-RSMT reduces wire length of 09.97% when compared with standard model [25] which is shown in Table IV. Secondly, experiment are carried out using ISPD98 benchmark [5] constructed by IBM as shown in Table V to evaluate performance of MDA-RSMT over standard routing method [8] considering computation time and memory utilization. From table VI it can be seen an average computation time reduction of 39.99% is achieved by MDA-RSMT model over FLUTE [14]. Similarly, experiment is conduced to evaluate memory utilization of MDA-RSMT and FLUTE using Valgrind [26] and [27] which is depicted in Table VII. From result obtained it can be seen an average memory utilization improvement of 90.63% is achieved by MDA-RSMT model over FLUTE [14]. From overall result attained it can be seen proposed MDA-RSMT model attain efficient performance considering utilizing memory efficiently, minimizing wire length and computation time (delay). Thus, reduce power or energy consumption of large and dynamic VLSI circuit.

TABLE I. BENCHMARK USED FOR EXPERIEMNT ANALYSIS CONSIDERING FIXED BLOCKS SIZE

Benchmark Case	Number of Pin	Number of Blocks
RC01	10	10
RC03	50	10
RC05	100	10

TABLE II. WIRE LENGTH (WL) PERFORMANCE ANALYSIS CONSDERING FIXED BLOCK SIZE

Benchmark Case	Existing Approach Wire length (um) [11]	MDA-RSMT Wire length (um)
RC01	29140	25256.3
RC03	62270	49021.4
RC05	87320	77018.4
Average	59576.667	50432.033

TABLE III. BENCHMARK USED FOR EXPERIEMNT ANALYSIS CONSIDERING DIFFERENT BLOCKS SIZE

Benchmark Case	Number of Pin	Number of Blocks
RC06	100	500
RC08	200	800
RC09	200	1000

TABLE IV. WIRE LENGTH (WL) PERFORMANCE ANALYSIS CONSDERING DIFFERENT BLOCK SIZE

Benchmark Case	Existing Approach Wire length (um) [11]	MDA-RSMT Wire length (um)
RC06	94742	83101.4
RC08	137116	123850
RC09	149274	136192
Average	127044	114381.13

TABLE V. BENCHMARK DETAILS

Benchmark Circuit Case	Number of Net	maximum degree	Average Degree
IBM1	14111	42	3.58
IBM13	99666	24	3.58
IBM18	201920	66	4.06
Average	105232.33	44	3.74

TABLE VI. COMPUTATION TIME PERFORMANCE

Benchmark Circuit Case	MDA-RSMT	FLUTE [14]
IBM1	89000	180000
IBM13	400000	570000
IBM18	1089000	1880000
Average	**526000**	**876666.66**

TABLE VII. MEMORY UTILIZATION PERFORMANCE

Benchmark Circuit Case	MDA-RSMT	FLUTE [14]
IBM1	108260	398908
IBM13	650232	2551928
IBM18	162422	6982462
Average	**306971.33**	**3311099.33**

IV. CONCLUSION

From extensive survey it can be seen modern circuit consist large of obstacle. As a result incurs routing delay and high amount of resource and power consumption. For addressing, this work presented a novel over the block routing for RLC VLSI circuit. However, placing on top of block is not allowed. Thus, placing buffer as close to the block will aid in utilizing resource efficiently and minimize power consumption. However, it is challenging as it requires accurate measurement model for meeting delay requirement of modern large VLSI circuit. This work presented such model namely, MDA-RSMT that meets memory and delay requirement of modern VLSI circuit. MDA-RSMT reduces amount of resource (buffers), wire length and computation time. Experiment are conducted on large and dynamic VSLI circuit. The outcome attained shows MDA-RSMT reduces wire length size by 14.35% and 09.97% considering without and with obstacle. Along with improves memory utilization by 90.63% and reduces computation time by 39.99% over standard routing model. From overall result attained it can be seen MDA-RSMT model attain efficient performance considering utilizing memory efficiently, minimizing wire length and computation time (delay). Thus, reduce power or energy consumption of large and dynamic VLSI circuit. The future work would further, consider minimize routing delay and memory overhead. Along with consider performance evaluation considering different benchmarks.

REFERENCES

[1] B.S. Gill and D.S. Modha. Sarc: Sequential prefetching in adaptive replacement cache. In Proceedings of the 2005 USENIX Annual Technical Conference, 2005.

[2] N. Megiddo and D.S. Modha. Outperforming LRU with an adaptive replacement cache algorithm. Computer, 37(4):58 – 65, 2004.

[3] M. Lukac, B. Shuai, M. Kameyama and D. M. Miller, "Information-Preserving Logic Based on Logical Reversibility to Reduce the Memory Data Transfer Bottleneck and Heat Dissipation," 2011 41st IEEE International Symposium on Multiple-Valued Logic, Tuusula, 2011, pp. 131-138.

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

[5] 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.

[6] Robert Wille, Anupam Chattopadhyay, Rolf Drechsler, "From reversible logic to quantum circuits: Logic design for an emerging technology", Embedded Computer Systems: Architectures Modeling and Simulation (SAMOS) 2016 International Conference on, pp. 268-274, 2016.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] K. Ma, Q. Zhou, Y. Cai, C. Zhang and Z. Qi, "A Steiner tree construction method for flexibility and congestion optimization," 2013 International Conference on Communications, Circuits and Systems (ICCCAS), Chengdu, 2013, pp. 519-523.

[13] 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.

[14] 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.

[15] C. J. Alpert, A. Devgan, and S. T. Quay, "Buffer Insertion for Noise and Delay Optimization," IEEE Trans. on Comput.-Aided Des. Integr. Circuits Syst., vol. 18,no. 11, pp. 1633–1645, 1999.

[16] T. Huang and Evangeline F.Y. Young. An Exact Algorithm for theconstruction of Rectilinear Steiner Minimum Trees among Complex Obstacles. In Proc. DAC, 2011.

[17] L. Li, Z. Qian, and Evangeline F.Y. Young. Generation of OptimalObstacle-avoiding Rectilinear Steiner Minimum Tree. In Proc. ICCAD, 2009.

[18] L. Li and Evangeline F.Y. Young. Obstacle-avoiding Rectilinear Steiner Tree Construction. In Proc. ICCAD, 2008.

[19] Huang T, Li L, Young E F Y. On the construction of optimal obstacle-avoiding rectilinear Steiner minimum trees.Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, 30(5): 718-731, 2011.

[20] C. H. Liu, S. Y. Kuo, D. T. Lee, C. S. Lin, J. H. Weng and S. Y. Yuan, "Obstacle-Avoiding Rectilinear Steiner Tree Construction: A Steiner-Point-Based Algorithm," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 7, pp. 1050-1060, July 2012.

[21] Yen-Hung Lin, Shu-Hsin Chang and Yih-Lang Li, "Critical-Trunk-Based Obstacle-Avoiding Rectilinear Steiner Tree Routings and Buffer Insertion for Delay and Slack Optimization," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 9, pp. 1335-1348, Sept. 2011.

[22] S. Hu, C. J. Alpert, J. Hu, S. Karandikar, Z. Li, W. Shi, and C. N. Sze, "Fastalgorithm for slew constrained minimum cost buffering," in Proc. Design Automation Conf., pp. 308–313, 2006.

[23] Zhang H, Ye D Y. Key-node-based local search discrete artificial bee colony algorithm for obstacle-avoiding rectilinearSteiner tree construction. Neural Computing and Applications, 2015, 26(4): 875-898.

[24] Hao Zhang, Dong-yi Ye and Wen-zhong Guo, A heuristic for constructing a rectilinear Steiner tree by reusing routing resources over obstacles, Integration, the VLSI Journal, 2016.

[25] Yen-Hung Lin, Shu-Hsin Chang and Yih-Lang Li, "Critical-Trunk-Based Obstacle-Avoiding Rectilinear Steiner Tree Routings and Buffer Insertion for Delay and Slack Optimization," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 9, pp. 1335-1348, Sept. 2011.

[26] J. Seward and N. Nethercote, "Using Valgrind to detect undefined value errors with bit-precision," in Proc. of the USENIX Annual Technical Conference, 2005, pp. 2–2.

[27] N. Nethercote, R. Walsh and J. Fitzhardinge, ""Building Workload Characterization Tools with Valgrind"," 2006 IEEE International Symposium on Workload Characterization, San Jose, CA, 2006, pp. 2-2.