COMPARISON OF ROUTING ALGORITHMS ON VLSI FLOOR PLANNING

[1]G.Revathy, [2]Dr.Abdul Razak

[1, 2] Department of Electrical Engineering,

Birla Institute of Technology and Science-Pilani (Dubai Campus), United Arab Emirates.

ABSTRACT

This is an era of VLSI where we are able to witness greatest miniaturization with respect to area and power. One of the major steps in miniaturization happens in physical design and particularly in VLSI floor planning and placement of blocks. The placement needs to be in such a way that they occupy very less area and leads to a compact design. Another challenge in this step is establishing interconnections between the blocks which is called routing. Thus, routing plays a major role in physical design. There are two different types which include single constraint and multi constraint routing. There are also many different ways where we could route to reach an optimum design, but the best way must be known and this can be done only by means of comparison. Hence a comparison must be made between two different algorithms so as to know the tradeoffs between them and their advantages of one over the other. The comparison is again referred with respect to previous literature for bench marking.

Keywords: ***Dog Legs, Floor Planning, Multi Constraint, Routing Algorithms, Single Constraint, VLSI***

I. INTRODUCTION

So many works were proposed in order to reduce the area consumed for VLSI floorplanning, and one amongst them is reducing the layout space by means of taking steps during routing itself, as wiring also consumes considerable amount of space. This chapter will discuss the existing works that have been proposed and their formulated algorithms to reduce area and decrease interconnect delay from source to sink. Zhou et al [1] considers buffer placement problem where during global routing, macro blocks permits wire to pass, but does not allow buffer insertion. Hence, [1] take buffer restrictions in account and solve this problem. Then given restricted constraint buffer and source, sink pins, a polynomial time exact algorithm is used to find route with buffers between two points having the lowest delay [1]. But, this will be complex when compared to traditional approach. Hence, it was improved by look ahead concept [2].

[2] describes a algorithm to route interconnect based on grid graphs. S-RABILA constructs maze routing path by taking obstacles of wire and buffer so that IC delay is reduced. A new look ahead technique is used to fasten the execution time of algorithm and also, it helps to find a proper solution. This could provide improvisations in performance over already proposed routing algorithms. One of the earliest literatures that implemented Swarm Intelligence in VLSI was Chen Dong[3]. It presents application of discreet particle swarm optimization (DPSO) to solve Minimum Rectilinear Steiner Tree problem and it proposes improvised routing algorithm based on Discrete particle swarm optimization. Adjustments are made in the parameters to find minimum rectilinear Steiner tree and achieve IC for existing nodes in the circuit. But, the disadvantage is that it does not have any

buffer or obstacle inclusion in the graph. Realizing the need of considering buffer placement and wire sizing, Ayob et al[4] proposed use of PSO in modeling problems in[3] with additional consideration of buffer placement in improving time delay of VLSI routing. Particle swarm optimization is a robust methodology based on how swarms move and share information. The technique of how doglegs is located is used to model problem for routing. [5-6] Xing She Yang introduced Firefly Algorithm (FA) for VLSI routing to solve complex combinatorial problem This algorithm is inspired by flashing behavior of fireflies. Fireflies flash light from their bodies to attract other fireflies and fireflies with relatively higher height intensity will result in attraction of other fireflies towards it. This lifestyle of fireflies is observed and changed into optimization algorithm. The light intensity represents fitness of solution and movement of fireflies represents improvising mechanism of agents. Buffer insertion [11] and wire sizing [12] helps to reduce IC delay between two points. [11] Chooses buffer position for wiring a tree such that Elmore delay is minimum. The buffer is chosen in such a way that the time for departure at source is as late as possible. The algorithm uses Depth First Search algorithm to construct time/capacitance pairs that correspond to different positions. [12] Presents algorithm for timing optimization by wire sizing. They reduce cost function by given time constraints with power dissipation reduce, they also do area minimization. In [13] , the authors makes use of delay optimization for achieving timing constraints which in turn helps perform VLSI circuits better. [14] Places buffer simultaneously along with optimized delay. But, the disadvantage is that it is restricted to routing using one constraint only. [15] shows that time constraints should also be included along with buffer insertion. It excels in both runtime and solution quality. It works under dynamic programming framework. [16] Is used to solve multi constraint interconnect routing problem. We make use of multi constraint routing algorithms for the following reasons:

- NP complete behavior seems only to occur in specially constructed graphs.
- Exact algorithm that are equally complex as heuristics in algorithm structure and in rub time on topologies that do not induce NP complete behavior
- Restricting number of k-paths explored during path computation.

Principle of non-dominance [16] reduces search space. But, multi constraint routing algorithm requires more simulation time compared to single constraint routing [16]. There are basically 2 techniques involved for global routing. One is Concurrent [17] and the other one is sequential [18]. In the former technique, as VLSI circuit routing problem is large, based on distribution of nets, the area of chip should be iteratively cut into small regions till it can efficiently dealt. Then, successively we paste the adjacent region together obtain routing of whole chip. Sequential routing is used to find detailed route by considering multiple constraints like power, skew, delay, area, etc [19]. Maze routing algorithms is used to route source to sink nets in a grid graph [20]. They find the cheapest cost path between two points before buffer insertion. The routing area is represented by 2D graph where wire and buffer location are specified with source and destination vertices. Most buffer insertion algorithm avoids the impact of inductance. [22] Propose new algorithm for RLC buffer insertion. A new technique for pruning is used to accelerate and estimate frequency for calculating the delay.

II. DELAY MODEL

The delay model to estimate interconnect delay has evolved for lumped RC model [23] to sophisticated high order moment matching delay model [24]. Basically, for current VLSI technology, the IC net can be visualized as distributed RC network [15]. For exact computation, RLC model can be used [24], [25]. But, it is more complex. For a routing problem to be analyzed well, the delay has to be calculated and delay calculation can be described in three separate parts: delay model and Elmore delay calculation. This delay calculation is common for both single constraint and multi constraint routing and the Elmore delay can be recursively calculated by programming the algorithm dynamically [24].

2.1 Delay Model

It is a known fact that as wire sizing decreases, resistance of the wire increases and it affects the interconnect delay. To understand the delay mechanism, IC is modeled as distributed Resistor-Capacitor network [6-7] and the delay from source to sink is calculated by applying Elmore delay. In Elmore delay, a π model is used to represent a wire segment as an RC circuit.

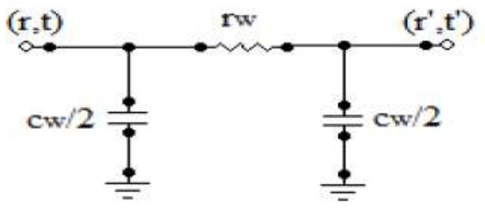


Figure 1 Wire Model [8]

Fig.1 illustrates π model where r is the resistance of the wire and c is the capacitance of the wire. The IC delay can be improvised by adding buffers at certain locations in the wire. Fig. 2 illustrates buffer included model, where d_b, r_b and c_b are intrinsic buffer delays.

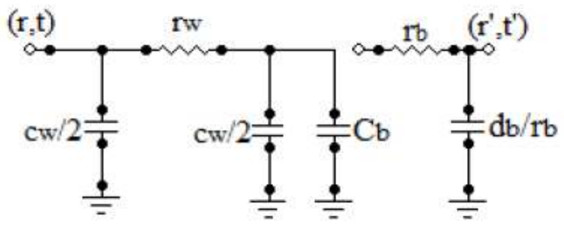


Figure 2 Wires With Buffer Model

2.2 Elmore Delay Calculations

For delay computation, each node in the wire is labeled with resistance delay pair (Ω,t) where Ω is resistance of wire and t is the delay associated upto that node[1] . When (Ω,t) is known for a node, the subsequent (Ω', t') can be calculated using:

$\Omega' = \Omega_w + r$ (1)

$t' = (r + \Omega_w/2)\, c_w + t$ (2)

where Ω_w is the resistance and c_w capacitance of the wire segment. It is calculated using Elmore delay formula. The formula for next adjacent pair of (Ω',t') is :

$\Omega' = \Omega_b$ (3)

$t' = \Omega\,(c_w + c_b) + \Omega_w\,(c_w/2 + c_b) + d_b + t$ (4)

where d, Ω_b and c_b are buffer delays, resistance and capacitances respectively.

III. AN OVERVIEW OF SINGLE CONSTRAINT ROUTING ALGORITHM

There are so many algorithms formulated for single constraint routing and one such algorithm is the firefly algorithm [8]. This algorithm is applied in modern VLSI routing and is inspired by flashing behavior of fireflies, as mentioned before. To understand firefly algorithm in a better way, grid graph model is to be known.

3.1 Grid Graph Model

Fig 3 shows gray colored source and sink, shown as a dark block, the buffer obstacle shown as a gray block and wire obstacle shown as a dark block again[8]. The value of each vertex is shown in Fig 4. The value of z set to 1 indicated presence of wire, and 2 represents region where buffer is not permitted. Value 0 represents the area that can be used to route and place buffer in the location. The general representation of value of vertex is:

$$V(r,c) = z \qquad (5)$$

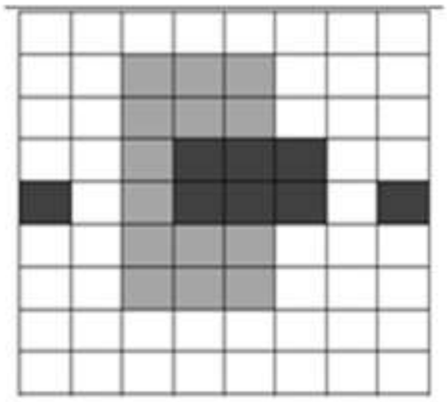
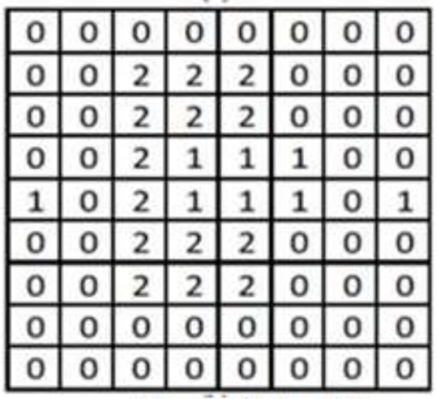


Figure 3 Location of Obstacle



0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0
0	0	2	2	2	0	0	0
0	0	2	1	1	1	0	0
1	0	2	1	1	1	0	1
0	0	2	2	2	0	0	0
0	0	2	2	2	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 4 Vertex Value

3.2 Modeling and Implementation

Fig 5 is taken into consideration. For each firefly, there exists a seven vector position from e_1 to e_7 [8]. Node location in the graph represented by this vector position is as illustrated in Fig 5, where e_{even} is position measurement in grid graph along x axis and e $_{odd}$ is position measurement in grid graph along y axis. On connecting them, the path between source and sink is established.

Source → e_1 → e_2 → e_3 → e_4 → e_5 → e_6 → e_7 → sink.

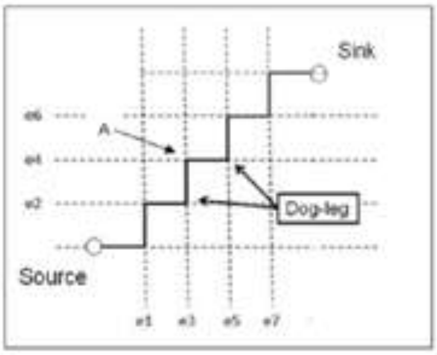


Figure 5 PSO VLSI Routing Mapping

A maximum of 8 doglegs are used to solve this particular case but as complexity increases, the number of doglegs can also be increased. The general representation of fire fly algorithm is shown in (6) [8]:

$$S_m = \begin{bmatrix} e1 \\ e2 \\ . \\ . \\ en-1 \\ en \end{bmatrix} \qquad \textbf{(6)}$$

Fig 6 is used to plot co-ordinates of the vertices[8]. The source So is (2,4) and sink co-ordinate Si is (7,6). Given that firefly representation is s = [3,3,4,2,7] , the path represented by this firefly is as in Fig 7.

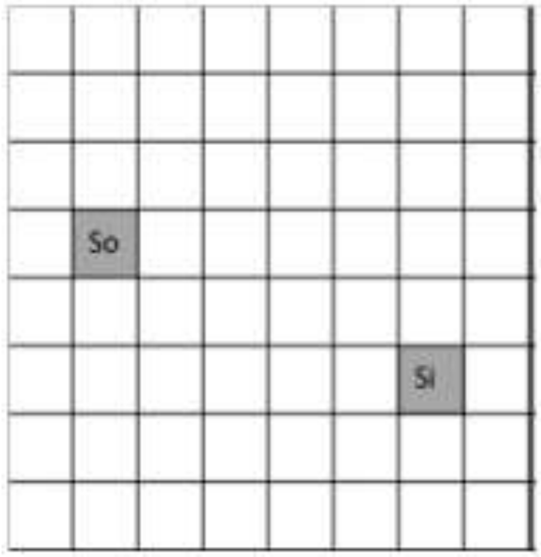


Figure 6 Location Of Source And Sink

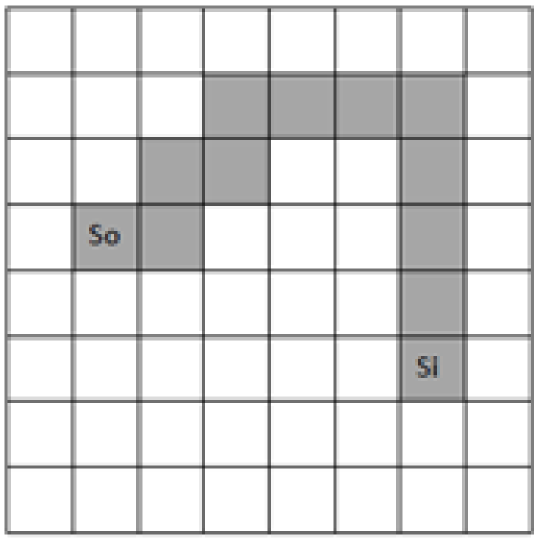


Figure 7 A Complete Path From Source To Sink

The number of segments is given by:

$$P = (\sum_{a=1}^{n} | ea - ea + 2|) + | Xsi - e\, n - 1 | + | Xso - e1 |) \qquad (7)$$

Where n = maximum no of doglegs

ea = location of a node in grid graph

X_{so} and X_{si} = x-axis coordinates for source and sink.

For every 6 segments, a buffer will be placed randomly [8].

The below given algorithm displays adaptation of the original firefly algorithm for routing.

Algorithm 1: Firefly algorithm for VLSI routing optimization problem[27]

- Set fitness function, f(s_m) according to (1) to (5) where s_m representation as (8)
- Generate randomly initial population of agent, s_m where m = 1,2,..,q
- Place the buffers randomly for population agent
- Find agent's light intensity, I_m at x_m using (1) to (5)
- Define light absorption coefficient, γ
- While z < t
- For m=1 to q
- For n = 1 to q
- If $I_m < I_n$
- Move agent m towards u using (12)
- Place the buffers randomly for agent m
- Perform correction if necessary
- Evaluate new solution using (1) to (5), update I_m using (11) and global best if necessary
- End if
- End for n
- End for m

- End while
- Post process results and visualization

Initially, in the algorithm, the fitness function has been defined. Its derived from (1) to (5). Then, the algorithm randomly generates initial population. The light intensity I_m is:

$I_m= 1/f(s_m)$ (8)

The iteration continues until stopping criteria is met, where stopping criteria will be a max. no of iteration, [8]. In every step, firefly will progress to larger light intensity firefly. That movement is determined by the below given equation:

$S_m = s_m + \beta_o e^{-\gamma r2mu} (s_m-s_u) + \alpha e_m$ (9)

where β_o is the factor of which it is attracted at r = 0

γ is coefficient of absorption,

α is randomization parameter in range [0,1]

e_m is a random number, a vector taken from uniform distribution

r is the Cartesian distance between fireflies

Given two fireflies a and b, the Cartesian distance between the two can be calculated from (10):

$r_{ab} = || s_a – s_b||$ (10)

After the movement of one agent towards next agent is made, the new position may be invalid due to approximate coordinate location. Eg, s_1 = [7.23, 1.2, 2.27] and s_2 = [3, 2.91, 1.1]. These solutions are round to integer values. So, s_1 = [7, 1, 2] and s_2 = [3,3,1].

After this, the upcoming stage will be to calculate fitness of new firefly and updating the new intensity of light. If new fitness is smaller, it is kept as the best solution[8].

3.3 Experimental Result

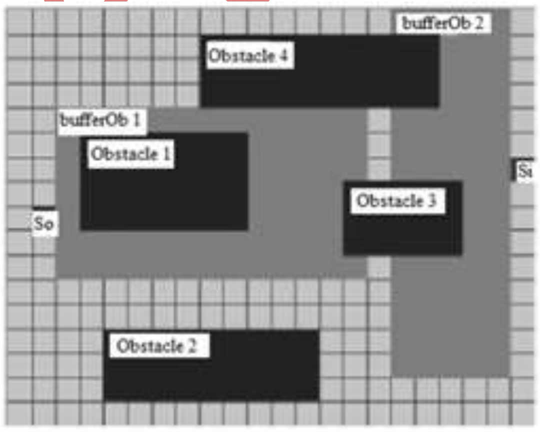


Figure 8 Case Study With 22 X 17 Grids

The above figure shows the case study from [4] while the below Table 1 gives the information about it. Visual basic 6 was used to program and parameters values used by several literatures and this study for solving the case study is listed in the following Table 2.

Table 1 Information Of The Case Study [8]

Resistance at sink	140 Ω
Load capacitance at sink	.002 pF
Resistance of wire segment	58 Ω
Capacitance e of wire segment	0.042 pF
Input capacitance of bugger segment	0.002 pF
Output resistance of buffer segment	140 Ω
Intrinsic delay of buffer segment	40 ps.

Table 2 PSO parameter comparison used in previous research with this study

	PSO	**FA**
Common parameters		
Number of agents, q	Unknown	10
Number of iterations, t	400	400
Number of computations	Unknown	5
PSO parameters		
Inertia weight, w	0.9 → 0.4	Not applicable
Cognitive component, c1	1.42	Not applicable
Social component, c2	1.42	Not applicable
r1 and r2	[0,1]	Not applicable
FA parameters		
Attractiveness, $_{\beta}$o	Not applicable	1
Randomization parameter, α	Not applicable	1
Absorption coefficient, $_{\gamma}$	Not applicable	1

Table 3 comparison of results obtained by PSO and FA

	PSO	FA
The least iteration number while global convergence	Not available	31 iterations
The average iteration number while global convergence	Not available	79 iterations
Optimal solution found	521.73 ps	521.73 ps

The result obtained by FA is similar to results obtained in [4]. The optimal solution was found to be 521.73ps in the current algorithm. Table 3 lists analysis obtained for future benchmarking.

IV. AN OVERVIEW OF MULTI CONSTRAINT ROUTING ALGORITHM.

This routing makes use of the look ahead paradigm [14]. The main aim of the [14] is to find the buffer insertion points and wiring sizes so that multiple constraints such as delay, no. of buffers etc are optimized. The input graph is represented in form of 2D grid graph, G is given by (V,E,W) in which $V = v_{src} \cup v_{snk} \cup v_n$ and E is the edge set. V_{src} is vertex corresponding to source , v_{snk} is the vertex corresponding to sink and v_n is the intermediate vertex. Any one of the vertices $v_i \in v_n$ can have chances of belonging to buffer obstacle set v_{ob}, or it can belong to wire obstacle vertices set v_{ow}. For each edge, $e(v_i, v_{i+1}) \neq v_{ow}$.

4.1 Graph Pruning

We should remove the redundant vertices v_i such that the space to be searched is pruned, and hence the effort in constructing routing path become less[9]. Thus, pruning is done when:

$$L(v_i \rightarrow v_{snk}) + L(v_i \rightarrow v_{src}) > L(v_{src} \rightarrow v_{snk}) \qquad (11)$$

Where $L(v_i \rightarrow v_{snk})$ = minimum path length from vertex v_i to sink,

$L(v_i \rightarrow v_{src})$ = minimum path length from vertex vi to source.

This path is permitted to pass through v_{ob}. A reference path is defined from source to sink, without passing through v_{ob} and an if no path exists, the value assigned will be infinity[9]. In path v_{ow} , wiring will not be permitted, and hence a path cannot pass through that area. Buffers are free to be inserted at any place once (11) is satisfied since reference path dodges v_{ob} .

4.2 Dominated Path

There are so many subpaths stored for v_i. (r,t) pair characterizes each vertex where r = resistance that was accumulated and t = time delay upto that particular vertex v_i from source vertex. If a new subpath is added, a new (r,t) is determined for v_i and the subpath is checked for dominance[9]. For any two resistance delay pairs, (r_1,t_1) and (r_2,t_2), we say the former dominates latter if $t_1 \geq t_2$ and $r_1 \geq r_2$ so that when it is proven to have redundancy, it can be pruned.

4.3 Look Ahead Concept

It was formulated in[14] to improve execution time. The distance between v_{src} and v_{snk} , which is the distance between source and sink of shortest path which dodges v_{ow} is first determined. A corresponding 1-D graph is created.

4.4 Path Traversal:

To solve the problem of multi constraint optimization, the three principles which were proposed in [16] could be adapted.

(i) Nonlinear measure for path length

(ii) A k-shortest path approach

(iii) Principle of non-dominated paths.

4.4.1 Non Linear Path Length Measure

A solution for multi constraint problem is by treating all non linear constraints within the constraints bound. All constraints are treated as equally significant and an important corollary non linear path length cannot be the shortest path necessarily, which suggests the use of k-shortest path approach[16].

4.4.2 K-Shortest Path Algorithm

It is an extension to Dijkstra's algorithm where it can return shortest path, the next shortest, the third shortest and so on till k-shortest paths. In SAMCRA[16], the technique of k-shortest path is put into use to node v_i that is intermediate on the path from two points to trace many sub paths from v_{src} to v_i. All sub paths are not saved so that there will be a search space reduction.

4.4.3 Principle of Non-Dominance

SAMCRA[16] can take into account only the sub path which is not dominant. This can reduce search space, and the condition is that , the path p_m is said to be dominated by path p_n if $w_l(p_m)>w_l(p_n)$ for $l = 1\ldots, |w|$, with inequality for at least one l. It can be relaxed by means of multi dimension.

4.5 Simulation

In simulation, we use two case studies for 22 x 17 and 80 x 40 graph sizes used in[14]. To verify, we use same PTM model[26] as in [14]. Load capacitance at sink vertex is 0.022pF, wire resistance is 37.5Ω, wire capacitance is 0.1026pF, input capacitance of buffer is 0.022pF, output resistance of buffer is 104.2Ω, and buffer delay is 20ps.

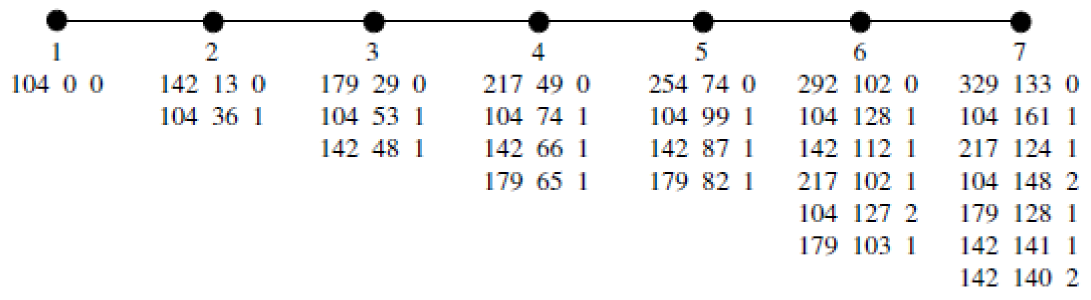


Figure 9 1-D Graph With Look Ahead Weight Vectors- All Are (R,T,#B) Pairs. Resistance R Is In Ω And T In Ps [9]

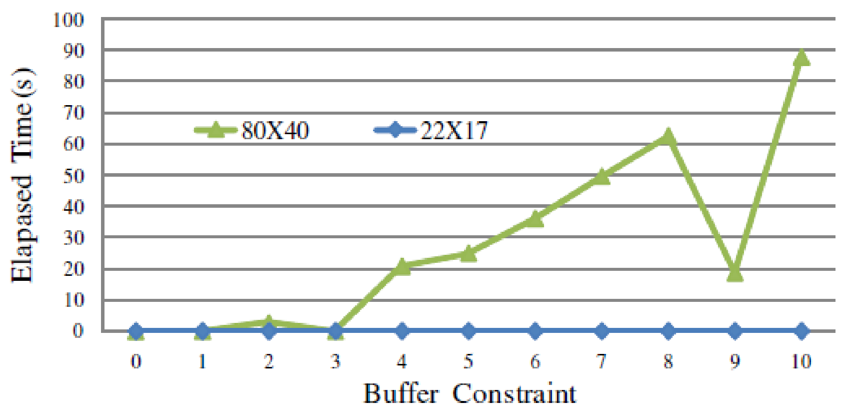


Figure 10 Elapsed Time For Path Traversal As Number Of Buffers Constraint Increases [9]

For the purpose of this work, we limit routing constraints to delay and no of buffer. Hence, routing uses (c,t,#B) pairs and look ahead (r,t,#B) pairs. Compared to [14], more storage is needed to keep value of (r,t,#B) as compared to (r,t) pairs. This technique can also be extended to different constraints by using different constraint model.

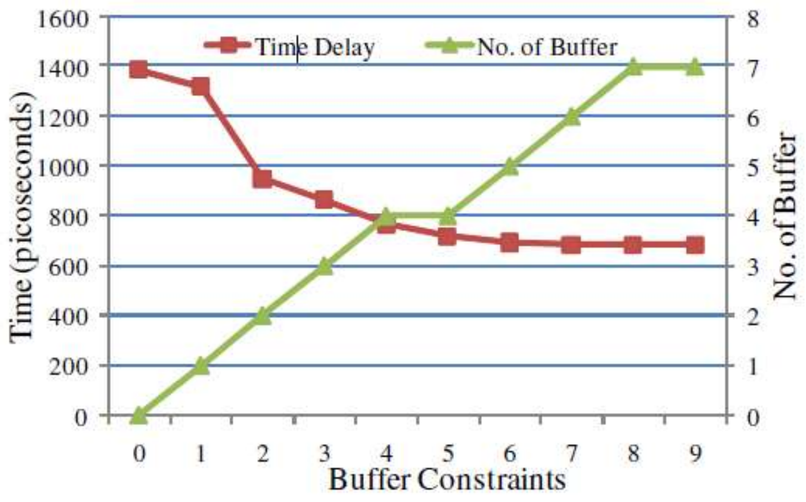


Figure 11 Multi Constraint Simulation Based On 22 X 17 Graph Size For The Delay Constraint Of 1400 Ps[9]

Fig 9 shows simulation results for routing 22 x 17 graph, we could notice the tradeoff between delay and number of buffer. We could fine tune sequential net routing by optimizing these constraints[9].

Table 4 Simulation Times For Three Important Parts Of Proposed Techinque. All Numbers Are Rounded To Nearest Tenth. The Time Percentage May Not Add Upto 100%[9]

Part of program	S-RABILA		MCRouting	
	Time (ms)	Time (%)	Time (ms)	Time (%)
Graph pruning	1.7	18.5	2.0	0.3
Look ahead	0.5	5.4	44.9	6.2
Delay calculation	7.0	76.1	676.0	93.5

Table 4 shows comparison of simulation times between S-RABILA[14] and proposed MCRouting algorithm. The delay calculation takes up majority of simulation time, and this shows that optimization should focus on this issue.

For delay calculation, path traversal makes up to majority of time simulation, Path traversal time increases almost exponentially as number of buffer constraint increases. Small elapsed time when buffer constraint is equal to 9 is due to pruning of patch search space[9].

V. CONCLUSIONS

The objective of firefly algorithm is to find minimum time delay path and placing buffers intelligently. Results obtained from its case studies show that proposed approach has potential for further extension. MCRouting is used to create routing paths through simultaneous wire sizing and buffer insertions. A look ahead method is used to estimate path lengths, and the search space can be reduced by non-dominance principle. We could optimize routing depending upon the selected routing constraints. Simulation for optimizing delays and no of buffer show that proposed technique could handle multiple routing constraints. However, it requires longer simulation time compared to single constraint routing although total time is in order of seconds for relatively small graph sizes. This scheme could be extended by using look ahead for global routing interconnect to solve single source multiple sink clock routing. Second is to propose more look ahead model such as for power, via and signal integrity for higher number of constraints.

VI. ACKNOWLEDGEMENTS

The authors thank Birla Institute of Technology and Science, Pilani for their support, from the Department of Electrical Engineering for the software support they have us unconventionally.

REFERENCES

[1] Zhou, D. F. Wong, I-Min Liu, A. Aziz (1999). Simultaneous Routing and Buffer Insertion With Restrictions on Buffer Location", in proceeding of the 36th Design Automation Conference, pp. 96-99, 1999.

[2] N. Shaikh-Husin and M. Khalil Hani, "Optimal Routing Algorithm for Minimizing Interconnect Delay", in the proceeding of the 2007 International Conference on Robotics, Vision, Information, and Signal Processing pp. 345-349, Nov 2007

[3] Chen Dong, Gaofeng Wang, Zhenyi Chen, Shilei Sun, and Dingwan Wang, "A VLSI Routing Algorithm based on Improved DPSO",Proceeding of IEEE International Conference on Intelligent Computing and Intelligent Systems, 20-22 November, Vol. 4, pp. 802-805, 2009

[4] M. Nasir Ayob, Zulkifli Md Yusof, Asrul Adam, Amar Faiz Zainal Abidin, Ismail Ibrahim, Zuwairie Ibrahim, Shahdan Sudin, N. Shaikh-Husin, M. Khalil Hani, "A Particle Swarm Optimization Approach for Routing in VLSI," cicsyn, pp.49-53, 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, 2010.

[5] Xin-She Yang, "Firefly Algorithm", Engineering Optimization: An Introduction with Metaheuristic Applications, pp 221-230, Wiley, 2010.

[6] Weste, N.H.E., and Harris, D. 2005 CMOS VLSI Design: A circuits and System Perspective. 3rd ed. Boston,MA: Pearson Education, Inc

[7] J.M Rabaey, A. Chandrakasan, B.Nikolic. 2003, DIGITAL INTEGRATED CIRCUITS: A Design Perspective. 2nd ed. Pearson Education 2012 International Symposium on Computer Applications and Industrial Electronics (ISCAIE 2012), December 3-4, 2012, Kota Kinabalu Malaysia

[8] M.Nasir ayob, Fariz Hassan, AHalim Ismail, H.Hassan Basri, M.Safwan Azmi, "A Firefly Algorithm Approach for Routing in VLSI", in the proceedings of 2012 International Symposium on Computer Applications and Industrial Electronics, Dec2012.

[9] Z.Ms-Yusof, M.Khalil-Hani, M.N.Marsono and N.Shaikh-Husin, "Optimizing Multi-Constraint VLSI Interconnect Routing", ISIC 2009.

[10] H. B. Bakoglu, Circuits, Interconnections, and Packaging for VLSI, Addison-Wesley, 1990.

[11] L. P. P. P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," in Proceedings of the International Symposium on Circuits and Systems (ISCAS 1990), pp. 865-868, 1990.

[12] J. Lillis, C.K.Cheng, and T. T. Y. Lin, "OptimalWire Sizing and Buffer Insertion for LowPower and a Generalized Delay Model," in Proceedings of the 1995 IEEE International Conference on Computer-Aided Design, pp. 138-143, Nov. 1995

[13] L. A. Glasser and L.P.J Hoyte, ”Delay and Power Optimization in VLSI circuits,” in Proceedings of the 21st Design Automation Conference, pp. 529-535, 1984

[14] N. Shaikh-Husin and M. Khalil-Hani, ”Optimal Routing Algorithm for Minimizing Interconnect Delay,” in Proceedings of the 2007 International Conference on Robotics, Vision, Information, and Signal Processing (ROVISP ’07), pp.345-349, Nov 2007

[15] Z. Jiang, S. Hu, J. Hu, and W. Shi, ”An efficent Algorithm For RLC Buffer Insertion” in Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED’07), pp. 171-175, March 2007.

[16] P.V. Mieghem and F.A. Kuipers, ” Concepts of Exact Qos Routing’Algorithms,” IEEE/ACM Transactions on Networking, vol.12, no.5, pp. 851-864, Oct. 2004

[17] T. C. Hu and M. T. Shing, ”A Decomposition Algorithm for Circuit Routing,” in VLSI Circuit Layout, T. C. Hu, and E. S. Kuh eds., IEEE Press, 1985, pp. 144-152.

[18] N. A. Sherwani, Algorithms for VLSI Physical Design Automation, 3rd ed., Kluwer Academic, Boston, MA, 1999.

[19] A. Nalamalpu and W. Burleson, ”A practical approach to DSM repeater insertion: Satisfying delay constraints while minimizing area and power,” in Proceedings of the 2001 IEEE ASIC/SOC Conference, pp. 152-156, Sep. 2001.

[20] E. F. Moor, ”The Shortest Path through a Maze,” Annals of the Harvard Computation Laboratory, pp. 185-292, 1959

[21] H. Zhou, D.F. Wong, I-Min Liu, and A. Aziz, ”Simultaneous routing and buffer insertion with restrictions on buffer locations,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.19, no.7, pp.819-824, Jul 2000

[22] Z. Jiang, S. Hu, J. Hu, Z. Li, and W. Shi, ”A new RLC buffer insertion algorithm,” in Proceedings of the International Conference on Computer- Aided Design, San Jose, California, pp. 553-557, 2006.

[23] J. Rubinstein, P. Penfield, Jr., and M. A. Horowitz, ”Signal Delay in RC Tree Networks,” IEEE Transaction On Computer-Aided Design, vol.2, no. 3, pp. 202-211, July 1983

[24] Y. I. Ismail, E.G. Friedman, and J. L. Neves, ”Equivalent Elmore delay for RLC trees,” Proceedings of the 36th Design Automation Conference (DAC’99), pp.715-720, Oct 1999

[25] Z. Md-Yusof, M. Khalil-Hani, N. Shaikh-Husin, M. N. Marsono, ”Iterative RLC Models for Interconnect Delay Optimization in VLSI Routing Algorithm,” in Proceedings of 2008 Student Conference on Research and Development (SCOReD 2008), Johor, Malaysia , 26-27 Nov. 2008, pp.270-1–270-4

[26] Predictive Technology Model (PTM). [Online] Available: http//www.eas.asu.edu/˜ptm.

[27] Ismail, M.M., Othman, M.A. , Sulaiman, H.A. , Misran, M.H. , “Firefly algorithm for path optimization in PCB holes drilling process”, Green and Ubiquitous Technology (GUT), 2012 International Conference , Jakarta, 7-8 July 2012