Improved Global Routing By Using A Star Algorithm

Abdulrahman Khalid*, Hossam Ahmed†, Mahmoud Mohamed‡ and Muhanad Atef§
Computer Engineering Dept., Faculty of Engineering, Cairo University
Cairo, Egypt
*abdulrahman.elshafei98@gmail.com, †hossamahmed201515@gmail.com, ‡mmmacmp@gmail.com, §muhanad.atef23@gmail.com

***Abstract*—In this paper VLSI routing is improved by improving global routing, this can be done by using A-Star with heuristic cost function that has parameters which affect the time taken by the router on changing instead of Dijkstra's algorithm in finding path, which will reduce time taken in this process and achieve the minimum wirelength, many comparisons are taken in this paper with different algorithms to find the optimum algorithm to be used to achieve both minimum wirelength and minimum time taken. From the comparisons of the paper we can find that using any algorithm is a trade off as when time taken is decreased, the wirelength is increased and vice versa, so there is no algorithm which is better from the other algorithms in general but using A-Star algorithm with the heuristic function in finding path is a good approach to be used in global routing as it decreases the routing time and achieve the minimum wirelength.**

***Index Terms*—VLSI Routing, Global Routing, Routing Algorithms, Fast Global Routing, Fast Routing Algorithms, Routing Algoritms Comparisons.**

I. Introduction

Routing is critical step in physical design process. Until now the optimum solution for VLSI routing has not been achieved yet, so it is considered as a very interesting challenging field. It is exactly done in two steps, global routing and detailed routing. At first global routing is run which is the responsible for making an approximate routing for the whole circuit in order to be used as a guide for detailed routing, then the detailed routing is run to make the exact routing for the system. That means if global or detailed routing is improved the whole routing process is improved, but there are many problems that have to be overcame to make a correct routing process. First of all the scale has to be taken into consideration, as millions of wires exist in a small chip area which means that many kilometers of wires are placed in a very small area, so wetotal wirelength has to be minimized as much as possible, also it is known that as the wirelength increases the resistance increases as well which means more delay in the chip. There is another problem as circuits are made in nano-scale which means that its geometric will be complex. Another problem is that routing algorithm have to be applicable for more than one layer with different costs. The direction of wires also have to taken into consideration as the direction of wires in every layer can be either vertical or horizontal and no diagonal paths, then to go from source 'S' to target 'T' the path taken should be in (vertical | horizontal) directions that specified by the layer (at each layer wires are placed in one direction only), then there is another problem as when a wire goes from layer to another to continue on the perpendicular direction it have to go through via which have a high resistance. DFM (design for manufacturer) rules also has to be achieved. All of these constraints must be taken into consideration with the global routing to achieve hundred percent of the circuit connections, that means global routing will take a lot of time to achieve all these constraints, and here is the challenge to achieve all the routing specifications with minimum time taken.

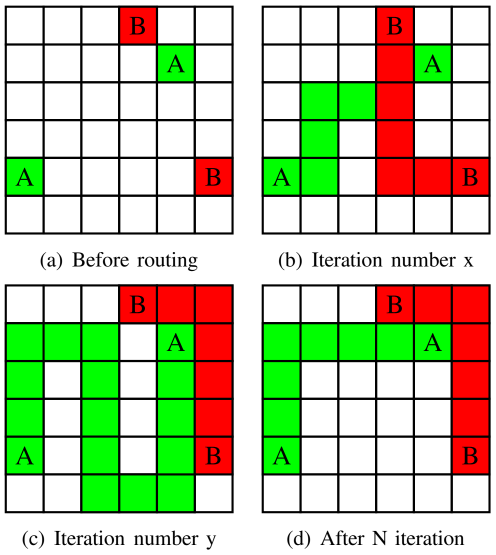

Fig. 1. Finding path in global routing

Figure 1 shows a very simple approach of how the global router works. At (a) the source and target of both (A,B) need to be connected ignoring obstacles, nevertheless we can observe that the global router have to iterate to get the best routing paths, at (b) (B,B) connected but there is no way to connect (A,A) as when (B,B) connected together they blocked the way for (A,A) to be connected, after some iterations we can find the figure at (c) in which (A,A) and (B,B) connected correctly, but there is a problem, the (A,A) connection is not the optimal path as there are paths which achieve less wirelength, so the global router have to iterate until reaching the optimal path. These iterations are done for only two connections in one layer without obstacles, then how about millions of wires in VLSI? this shows how much the global routing algorithm has to be very fast in order to connect this huge number of wires as fast as possible.

II. Related Work

Several papers proposed various types of approaches to improve the routing process, each of them tried to improve the overall routing by improving one or more parameters, some paperes tried to decrease number of vias, other papers tried to decrease the time taken and so on. In [2] they used a sequential global routing and they provided two bounded length maze algorithms as finding path algorithms in order to make the router faster and to avoid congestions thus avoiding overflow, the first one is optimal-BLMR and the second one is heuristic-BLMR. optimal-BLMR is used to get the minimum cost paths to be used as a routing paths, this can be done in three steps. First BLC (bounded length constraint) is defined as a grater number than Manhattan distance then to go from source to target the neighbour points are tested if it can be a part of path or not, each point that violates the BLC constraint is discarded. Second if the route started from point v and ended at point u and there was many paths between these two points, the normal maze algorithm wwill take the path with minimum cost which may cause the route get into congestions, bun in optimal-BLMR they keep track of all paths between these two points in order to choose the path that will not cause overflow, it iterates on the minimum cost path everytime and if it found a suitable path it reserves that path otherwise it discard that path. Third step the optimal-BLMR iterates on the reserved paths and choose the one to be used for routing. In heuristic-BLMR they tried to speed the router up by reserving only one path between the two points, but they also wanted to keep the advantage of optimal-BLMR (avoidng congestions) and this can be achieved by choosing the selected path only if the wirelength is enough to detour around congested regions. The advantages of this paper are using sequential global routing which is based on multithreaded global routing which speedup the router, avoiding collision by using optimal-BLMR, and making a fast and nearly avoiding collision algorithm (heuristic-BLMR). But there are some disadvanatges too, as optimal-BLMR is very slow to be used, although heuristic-BLMR is faster than optimal-BLMR its results are not accurate as the wirelength is not the best compared with other papers.

III. Experimental Analysis

we implement global routing with A_Star router algorithm in c++ language with Rsyn to parse input test cases files with format LEF/DEF. All experiments are perfromed on a Machine with 2 GHZ intel Processor and 8 GB RAM. our algorithm is focus on three important factors :total wire-lenght, channel congestion ,number of vias. Our target is to minimize a convex combination of the total wire-lenght and the number of vias that represent the bends in the tree that result in solving the net of pins to be connected.our test cases input are all from

TABLE I
Compare run time of Dijkstra router and A* router with Heurestic-Fucntion1

Design	A*				Dijkstra
	(0.2,0.1)	(0.3,0.3)	(0.5,0.5)	(0.5,0.5)	
test1	75.843764	87.797614	87.086193	96.377979	91.348007
test2	2.511599	2.606054	2.874605	2.497377	3.435606
test3	12.381436	14.959666	14.859677	14.766495	11.824573
test4	0.075472	0.073069	0.073769	0.072691	0.071829
test5	0.073988	0.072911	0.074165	0.072767	0.071945
test6	6.948016	6.266789	5.198969	6.906992	7.0385

TABLE II
Compare run time of Dijkstra router and A* router with Heurestic-Fucntion2

Design	A*				Dijkstra
	(0.1,0.1)	(0.2,0.1)	(0.3,0.3)	(0.5,0.5)	
test1	90.440165	93.288501	89.414646	96.377979	91.348007
test2	2.461400	2.469304	2.480592	2.497377	3.435606
test3	12.264829	13.449206	12.039944	14.766495	11.824573
test4	0.071577	0.157244	0.078624	0.072691	0.071829
test5	0.078316	0.090906	0.071333	0.072767	0.071945
test6	6.893310	6.400888	6.974920	6.906992	7.038505

ICCAD’19 contest for global routing.we test our code with different heurestic functions -that affect the run time of A*- to check its effect on CPU run time.our concern is on wire-lengh and number of vias result from routing without increasing infeasibility.we generate the trees of all nets in parallel to reduce running by using multi-threaded approach to generate the net results tree as this step has the most costly part of the router algorithm.

1) Experiment1: in this experiment we compare run time of router with dijkstra Router and out A* router with heuristic function with differents constants of heuristic functions that affect the run time of the router.heuristic function help us to know how near the point to the target point:

$Heurestic-Fucntion1 = (|p_1.x - p_2.x| + |p_1.y - p_2.y| * const1) + (|p_1.layer - p_2.layer|) * const2)$

$Heurestic-Fucntion2 = (ln(|p_1.x - p_2.x| + |p_1.y - p_2.y|) * const1) + (|p_1.layer - p_2.layer|) * const2)$

$Heurestic-Fucntion3 = (|p_1.x - p_2.x| + |p_1.y - p_2.y| * const1) + (ln(|p_1.layer - p_2.layer|) * const2)$

we notice from TableI,TableIII,TableII that heuristic function affect the running time of router ,decreases it to be lower than dijkstra router’s running time in many test cases. the running time changes with the change of heuristic function constants. we notice that higher constants increase the running

TABLE III
compare run time of Dijkstra router and A* router withHeurestic-Fucntion3

Design	A* Router				Dijkstra
	(0.1,0.1)	(0.2,0.1)	(0.3,0.3)	(0.5,0.5)	
test1	86.598509	83.204272	103.166426	102.378304	91.348007
test2	2.481847	2.486656	3.622759	2.556863	3.435606
test3	12.655603	15.447421	15.629251	20.004493	11.824573
test4	0.073731	0.070911	0.073468	0.103421	0.071829
test5	0.078171	0.079362	0.088107	0.099699	0.071945
test6	6.470339	7.195767	8.455288	7.637577	7.038505

TABLE IV
COMPARED WIRE LENGH,VIAS WITH TRIPLEZ AND NTUIDROUTE ALGORITHMS IN ICCAD'19 GLOBAL ROUTING CONTEST

Design	wire-length scores			vias score		
	TripleZ	NTUidRoute	ours	TripleZ	NTUidRoute	ours
test1	12485549.76	12992510.61	12150100	3261476.00	4017448.00	2803560
test2	569622.01	466809.64	389460	251728.00	318404.00	176728
test7	17800019.36	15935101.91	14663400	4265744.00	4789488.00	2666120
test3	2859212.04	2495661.79	2851900	695348.00	771328.00	446024
test8	74264784.88	65028116.52	70041500	19689760.00	23183264.00	12532800
Avg	21595837.61	19383640	20019272	5632811.2	6615986.4	3725046.4

time and sometimes be greater than dijkstra router's runtime. Heuristic function 2 has ovarall results which is better than others, with using more efficient heurestic functions,run time and wire-length and vias number combination may improve greatly.

A. Performance Comparsions

we compare results of our algorithm which focus on factors - total wire-lenght,number of vias in the routing results - with TripleZ and NTUidRoute algorithms in ICCAD'19 TableIV .threads number used in all cases is 8 threads. score used as unit to compare ,score of wire length is a function of metal pitch and wire length and weight:

$$WLScore = \frac{wire-length*weight}{metal-pitch}$$

score of vias is function of number of vias :

$$ViasScore = \#vias * 2$$

B. Observation

our algorithm has wire-length score less than TripleZ with 7.3% and greater than NTUidRoute with 3.28% but it affect the vias number greatly that our vias score is less than NTUidRoute's vias score by 43% which we try to decrease it as the vias is one of the sources of heat generation in the chip.

IV. CONCLUSION AND FUTURE WORK

The goal of this paper is to develop A-star-based global Router with minimum wire-length and number of vias as the more vias number the more heat generated from the chip with minimum run time.to overcome the time problem we use heurestic function that decrease the run time of router and test it with different constants to define its affect on run time ,for future work we aim to define effective heurestic function to improve the running time.using more effecient versions of A_star algorithm such as bi-directional A-star,iterative deeping A-star also improve the running time of out algorithm .for instance bi-directional A-star that processes edges in both directions that improve the running time as well as it save the memory demand for path saving as we will deal with both directions of edge.

REFERENCES

[1] Muhammet Mustafa Ozdal and M. D. F. Wong, "Archer: a history-driven global routing algorithm," *2007 IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2007, pp. 488-495, doi: 10.1109/ICCAD.2007.4397312.

[2] Liu, Wen-Hao, et al. "NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing." *IEEE Transactions on computer-aided design of integrated circuits and systems* 32.5 (2013): 709-722.