A Multi-layer Obstacles-Avoiding Router Using X-Architecture

Yu-Cheng Lin[1] Hsi-An Chien[2] Chia-Cheng Shih[3] Hung-Ming Chen[3]
[1]Dept. of Information and Electronic Commerce, Kainan University, Taoyuan, Taiwan
[2]Dept. of Information and Computer Engineering, Chung Yuan University, Chung-Li, Taiwan
[3]College of Electrical and Computer Engineering, NCTU, Hsinchu, Taiwan
linyu@mail.knu.edu.tw, hsianchien@gmail.com, maplume@gmail.com, hmchen@mail.nctu.edu.tw

Abstract: - In recent years, scaling down device dimension or utilizing novel crystallization technologies provide the opportunity of applying much more devices to integrated circuit fabrication. Due to emerging DSM effects, the research about routing has drawn much attention in VLSI Physical Design. In this paper, we will focus on three issues. One, the traditional Manhattan routing has longer length and larger delay than X-Architecture routing. Second, in multilayer routing, the delay of one via is much larger than the delay of Manhattan routing. Third, since a routed segment and macro cell should be considered as obstacles, we must consider the rectangle and non-rectangle obstacles, and consider the number of vias as well. Our algorithm can handle both rectangle obstacles and non-rectangle obstacles, and we use fewer vias and X-Architecture router by region to construct the multilayer routing trees. The main purpose is to obtain an obstacles-avoiding routing tree of minimal wire length and minimal delay.

Key-Words: - X-Architecture, routing, multi-layer, physical design, VLSI, algorithm

1 Introduction

In recent years, scaling down device dimension or utilizing novel crystallization technologies provide the opportunity of applying much more devices to integrated circuit fabrication. Therefore, Electronic design automation [6], which contains the placement and routing [9][10] steps, plays an important role to increase the digital IC designer's productivity and the novel technique is used for the digital and analogy design [22]. However, it also brings a lot of physical characteristics which were neglected in the past. The problems contain wire congestion, delay, crosstalk, etc. The research about routing has drawn much attention in VLSI Physical Design.

Over the past few years, a considerable number of studies have been made on routing and these studies of routing could be mainly divided into three different aspects. First, the majority of the researches mainly gave priority to shorten delay time as a result of minimizing the total wirelength [2][7][8][21]. Second, as the foregoing obstacles would block a routing fabrication, there are two ways to solve these obstacles at present. The first way is to obtain the routing result without considering the obstacles, and later to adjust the line among the obstacles [23]. Another way is to construct the routing tree considering the obstacles. It is called obstacle-avoiding rectilinear steiner minimal tree (OARSMT) [12]. There are many studies of OARSMT in the past, and the researches on single layer (2-D) have been drawn a lot of attention, such as [3][5][13][16][21]. Third, although a large number of studies have been made on 2-D routing tree, little is known about multi-layer routing tree. So far the technology of IC design has been able to meet the requirements for System in Package (SIP) or System on Package (SOP) [15][17][18][19][20]. However, the technology of multi-layer EDA is not mature enough. The problem of large wirelength still exists, and therefore it is necessary to improve the multi-layer routing.

Since the delay time of a via is larger than that of Manhattan style, and the routing of X-Architecture results in better wirelength, our algorithm uses less via numbers, and completes the routing by X-Architecture. First, appropriate location of via each layer must be found, and we divide the chip to four regions. Second, the routing of each region is completed by adopting Delaunay Triangulation (DT) [1] respectively. The spanning graph is applied to avoid the condition that the connection throughout the obstacle. Third, the shortest path is chosen from all connecting path, and the routing of the shortest path is completed by X-Architecture. The routings of all regions are constructed step by step until all of them are completed.

The routing strategy of avoiding non-rectangle obstacle is considerable additionally. It is discussed respectively that whether the pins are located on the routable region apart from the non-rectangle obstacle or not. As pin is inside the fictitious

rectangle, we construct the spanning graph additionally.

2 Previous Work

Because the technology of process progresses rapidly, there are more and more components can be put in the chip of same size. It will increase the complexity of routing problem and cause a lot of physical characteristics which were neglected in the past, hence routing becomes one of the key steps in physical design. There is a considerable number of studies have been made on routing, they can be classified into three main groups, we will introduce them separately in this section.

2.1 The X-Based Architecture Routing [7]

As technology advances into the nanometer territory, the interconnect delay has become a first-order effect on chip performance. To handle this effect, the X-Architecture has been proposed for high-performance integrated circuits. In [7], the authors presented the first multilevel framework for full-chip routing using the X-Architecture.

Since the optimal routing solution for each three-terminal net can be found easily, the authors used the Delaunay triangulation approach to divide all terminals into groups of three-terminal nets. After that, the authors computed the optimal wirelength of all three-terminal nets, and sorted them by their wirelength. Further, the authors iteratively picked up a group of three-terminal nets with the minimal wirelength, then routed and merged them to the X-Architecture Steiner Tree (XST) until it was constructed. Compared with the multilevel routing for the Manhattan architecture, experimental results showed that the method of this work reduced wirelength by 18.7 percent and average delay by 8.8 percent with similar routing completion rates and via counts.

2.2 Rectilinear Steiner Minimal Tree among Obstacles [23]

Rectilinear Steiner minimal tree (RSMT) is a fundamental problem in VLSI/ULSI physical design. Most of the works do not take obstacles into consideration. But in fact, macro cells, IP blocks, and pre-muted nets are often regarded as obstacles in the routing phase, even in placement and floorplanning.

[23] studied RSMT problem among obstacles and presented an $O(mn)$ 2-step heuristic for multi-terminal tree construction. Where m is the number of obstacles and n is the number of terminals.

2.3 Routing for Multi-layer Structure [17][18]

3D packaging via System-On-Package (SOP) is a viable alternative to System- On-Chip (SOC) to meet the rigorous requirements of today's mixed signal system integration. In [18], the author presented the first physical layout algorithm for 3D SOP that performs thermal-aware 3D placement and crosstalk-aware 3D global routing.

The 3D router is divided into the following steps: (a) coarse pin distribution (b) net distribution (c) topology generation (d) layer assignment (e) channel assignment, and (f) pin assignment. In the coarse pin distribution step, which is done before net distribution, the author find a coarse location for the pins and use this information for the net distribution. After the net distribution, the detailed pin distribution step assigns finer location to all pins in each routing interval. A Steiner tree based routing topology for each net is constructed and a layer pair is assigned to it during the topology generation step. The conflict among the nets for routing resources is resolved and layer pairs are assigned during the layer assignment step. The channel assignment problem is to assign each pin in the pin distribution layers to a channel in the placement layers. The purpose of pin assignment is to finish connection between the pins in the routing channel and the pins along the block boundary. As the author focused on analyzing the pin location and obstacles each layer, therefore, the author obtained the better wirelength and fewer vias.

3 Problem Formulation

It is well known that the traditional algorithm is generally used to make the connection between two pins with 2-D structure. Different from the traditional algorithm, we accomplish the connection between multiple pins with 3-D structure at the same time. First, we need to provide the algorithm with the initial source, plenty sinks, coordinates of the obstacle and number of the layers. Next, we avoid the unnecessary connection to achieve the connection between the initial source and sinks with our algorithm. At last, we obtain a routing tree with multi-layer and the minimal delay time.

Owing to the advantage of X-Architecture, we decide to adopt the method of X-Architecture to

reduce the total wirelength. There are only vertical and horizontal connections in the traditional Manhattan structure. However, we allow 45 degree connections under X-Architecture. Therefore, we will efficiently obtain a better wirelength with X-Architecture than that with the traditional Manhattan structure, such as Fig. 1. Besides, we will construct fictitious rectangles to solve the nonrectangular obstacles, and discuss the possible problem as the terminal inside the fictitious rectangle, such as Fig. 2.

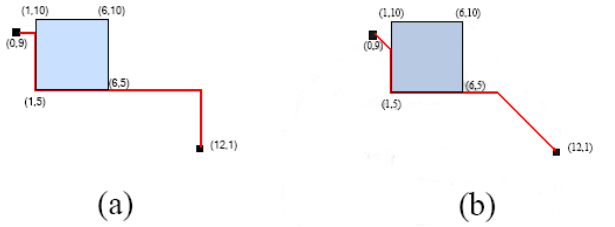

Fig. 1 (a) The traditional Manhattan steiner path. (b) The X-Architecture routing path.

Fig. 2 A terminal inside the fictitious rectangle obstacle.

4 Our Approach

Since constructing a routing tree is a complicated problem, the scholars hope to find an efficient algorithm to obtain a better routing. Besides the traditional Manhattan routing, the technology of X-Architecture is of great help in wirelength reduction. Therefore we can adopt the method of X-Architecture to improve the wirelength. In consideration of the obstacle-avoiding problem, it is necessary for the router to be able to deal with both the rectangular and non-rectangular obstacles. In multi-layer system, the delay time of one via is much larger than that of Manhattan length. According to this reason, we minimize the number of vias. On the basis of the above-mentioned thoughts, we present an algorithm, which is able to construct the X routing tree under the condition with both the rectangular and non-rectangular obstacles. Further, we minimize the delay time between the preliminary point and the extreme point.

```
Algorithm: Rectangular/ Non-Rectangular Obstacles-Avoiding X-Architecture Router
           in Multilayer System
Input: A set of T= (t1, t2, t3, ... , tn), O= (o1,o2,o3, ... ,on), NO= (no1,no2,no3, ... ,non) with
       the source under a set of layers L= (l1,l2,l3, ... ,ln)
Output: A routing tree which minimizes total wire-length
begin
    for each layer
      Find the location of via
      Divide into four regions according to the via location
      for each region
        if there are two pins in the cluster
          Find the shortest path of two pins
        else
          Complete Delaunay Triangulation
          Find the MST
          if the MST edge through out the obstacle
            if there is non-rectangle obstacle
              Build fictitious rectangle then Spanning Graph
              Find the shortest path
            else
              Construct the Spanning Graph
              Find the shortest path
          else
    Xroute
end;
```

Fig. 3 The pseudo code of our algorithm.

For the multi-layer construction of the steiner tree, our algorithm consists of the following steps: First, according to the locations of the individual pins, we figure out the central location. And then we project the central location into every layer. Second, we judge whether the location of via between layer *l* and layer (*l*+1) is applicable (*l*=1,2,3,...,n-1). Supposing the location of via between layer l and 15 layer (*l*+1) is not applicable, we switch the via to the suitable location. Third, as based on the location of via, we divide each layer into four regions. Afterward we connect all the pins in each divided region in the direction of each layer's central location. Furthermore, we use X-Architecture to modify the original results of the routing. The algorithm is shown in Fig. 3.

4.1 The Location of Via

The delay time of one via is much larger than that of Manhattan length. According to this reason, we minimize the number of vias. In this step, we must decide the location of via first, and the method, which we use to determine the location of via, is as follows:

First, according to the locations of the individual pins, we figure out the central location. And then we project the central location into every layer. Second, we find the total obstacles on layer *l* and layer (*l*+1). Later, we judge whether the location of via between

layer l and layer (l+1) locates inside any obstacle or not. (l =1,2,3,...,n-1). Third, supposing the location of via between layer l and layer (l+1) is not applicable. After finding out all the applicable locations, we separately calculate the lengths between the original central location and all the applicable locations. Then we pick out the suitable location closest to the original central location. Based on the location of via, we divide each layer into four regions.

4.2 The Method of Routing

In order to lower the complexity, we divide each layer into four regions according to the location of via in this thesis. Here we construct the routing of four regions on one layer in turn, and then we build the routing of the next layer in the same way, and so on until we complete the whole routing. For the routing of single region, we use the spanning graph to accomplish the regional connection and further adopt the Delaunay Triangulation (DT) [1] algorithm to figure out the shortest connection between three points. As we can get a better result by combining the spanning graph [21] and DT algorithm, the method of our routing is as follows:

First, we have to calculate the number of the points in single region. If the number is 1, we just do nothing. If the number is 2, we use the spanning graph to find the shortest path. If the number of the points is equal or greater than 3, we use the Delaunay Triangulation (DT) algorithm to obtain the result of the regional connection. With the result of the routing, which is brought by the DT algorithm, we use Kruskal algorithm to figure out the path of spanning tree. The Kruskal algorithm is to arrange all the edges and look for the shortest edge each time and put it in the tree. However, the edges are unable to lead into a loop. Thus we can obtain a shorter spanning tree.

In order to get a minimal spanning tree, we look for the unnecessary edges of the spanning tree, which is mentioned before, and remove them. The unnecessary edges mean that the terminal point of the edge is in one corner of the obstacle. Fig. 4 shows an example to explain the above-mentioned method. Fig. 4 (a) shows the points in the divided regions. Fig. 4 (b) displays the original routing in one divided region by using the DT algorithm. We get a shorter spanning tree with the result of the routing, which is brought by the DT algorithm, as shown in Fig. 4 (c). And we remove the unnecessary edges in Fig. 4 (d). Finally, we transform the MST into X-Architecture, as shown in Fig. 4 (e).

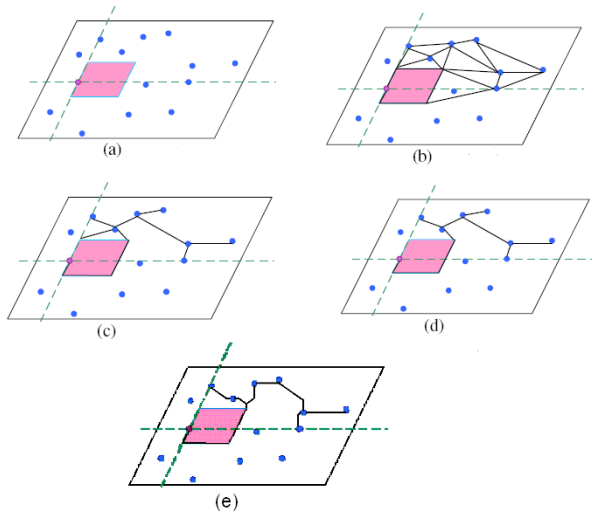

Fig. 4 (a) The points in the divide region.
(b) Original routing by using DT.
(c) Find the shorter spanning tree from DT.
(d) Remove the unnecessary edge.
(e) Xroute result.

4.3 Dealing with the Edge across Obstacles

Regardless of the range of the obstacles, the Delaunay Triangulation (DT) algorithm only considers the locations of individual points, including pins and corners of obstacles. Since the DT algorithm invariably forms some inappropriate edges across the obstacles, we construct the spanning graph to adjust such edges of the MST. However, we may divide the obstacles into two types as rectangular and non-rectangular obstacles. If there are rectangular obstacles, the spanning graph can be generated directly. Nevertheless, if there are non-rectangular obstacles, we may not obtain the spanning graph immediately. Therefore, we discuss the two above-mentioned problems separately in this section.

4.3.1 Rectangular Obstacles

If the edge of the MST is on the location across the rectangular obstacle, we call the edge as the inappropriate edge. First, we pick any point of the inappropriate edge as a preliminary point, and then pick another point as a terminal point. At First, we pick any point of the inappropriate edge as a preliminary point, and then pick another point as a terminal point. Next, we construct the spanning graph between the two points, moreover, we figure out the shortest path from the spanning graph. Finally, we transform the shortest path into X-Architecture. Here is an example, as shown in Fig. 5.

Fig. 5(a) displays the result by using the DT algorithm in the region. In Fig. 5(b), we obtain the MST from the result, which is generated by the DT algorithm. Afterwards, we pick out the inappropriate edge, which is marked as a red one in Fig. 5(c). And in this example, the two points of the inappropriate edge are just located at the boundaries of the obstacle. In Fig. 5(d), we regard one point as a preliminary point, and another point as a terminal point to re-construct the spanning graph with the corners of the obstacle.

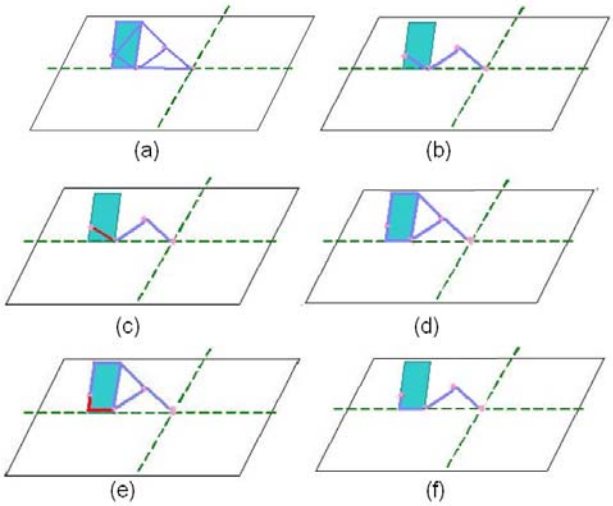

Fig. 5 (a)The DT result of the region.
(b)Find the MST from DT.
(c)Find the illegal edge.
(d)Build spanning graph to reroute the edge.
(e)Choice the shortest path.
(f)Xroute the shortest path.

Since the two points of the inappropriate edge are just located at the boundaries of the obstacle, we obtain two routing paths with the spanning graph. In Fig. 5(e), we select the shortest path to put into the tree. At last, we obtain a new minimal spanning tree, as shown in Fig. 5(f).

4.3.2 Non-Rectangular Obstacles

As the edge of the MST is on the location across the non-rectangular obstacle, we generate the additional edges to change the non-rectangular obstacle into a rectangular obstacle, which is called a fictitious rectangular obstacle. In this part, we give an example of taking an isosceles right triangle as a non-rectangular obstacle and two terminals.

First of all, we discuss the terminal location of the fictitious rectangular obstacle. If the terminal is outside of the fictitious rectangular obstacle, we use the spanning graph by regarding the fictitious rectangular obstacle as a normal rectangular obstacle. If the terminal is outside of the fictitious rectangular obstacle, we use the spanning graph to deal with the fictitious rectangular obstacle as a normal rectangular obstacle. However, if the terminal is inside of the fictitious rectangular obstacle, we need to construct the spanning graph within the routable region of the fictitious obstacle, and then generate the spanning graph outside of the fictitious obstacle. The detailed method will be described as follows.

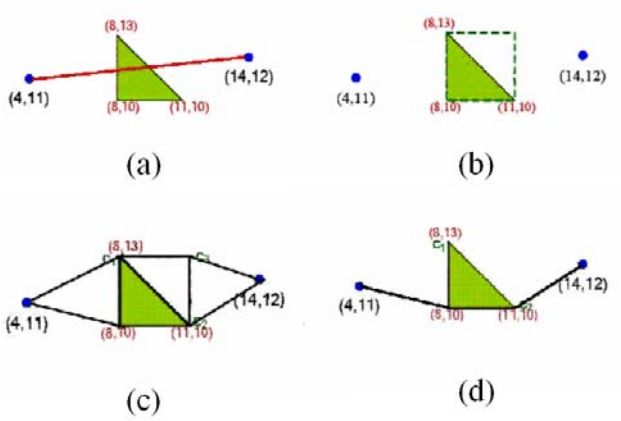

Fig. 6
(a) The edge through out the obstacle is called illegal edge.
(b) Build the fictitious rectangle and two pins are outside.
(c) Generate the spanning graph of the fictitious rectangle.
(d) Find the shortest path from the spanning graph.

First, we consider the case of terminal is outside the fictitious rectangle obstacle. Fig. 6(a) shows the inappropriate edge of the MST across of the obstacle. After forming the fictitious rectangular obstacle, both the two pins (4,11) and (14,12) are just outside of the fictitious rectangular obstacle, as shown in Fig. 6(b). With the fictitious rectangular obstacle and the two pins, we generate the spanning graph and put the triangular hypotenuse into the spanning graph, as shown in Fig. 6 (c). After that, we pick out the shortest obstacle-avoiding path and obtain the final result. Fig. 6 (d) shows that the path (4,11)-(8,10)-(11,10)-(14,12) is the result of this example. At last, we transform the shortest path into X-Architecture.

Then, we will discuss the case of terminal is inside the fictitious rectangle obstacle. After forming the fictitious rectangular obstacle, the pin (10,12) is just inside the fictitious rectangular

obstacle, as shown in Fig. 7(a). First, we construct the spanning graph with the corners (c1 , c2 , c3) of the fictitious rectangular obstacle and the pin (10,12) , as shown in Fig. 7(b). Afterwards, we form the spanning graph with the fictitious rectangular obstacle and another pin (4,11) , and put the 23 triangular hypotenuse into the spanning graph, as shown in Fig. 7(c). After that, we pick out the shortest obstacle-avoiding path and obtain the final result. Fig. 7(d) shows that the path (4,11)-(8,13)-(10,12) is the result of this example. At last, we transform the shortest path into X-Architecture.

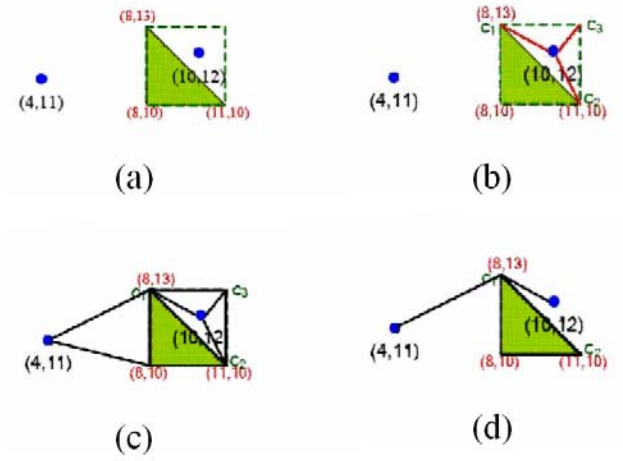

Fig. 7
(a) Build the fictitious rectangle and pin (10,12) is inside the fictitious rectangle obstacle.
(b) Build the inside spanning graph of pin (10,12) and the fictitious.
(c) Generate the outside spanning graph of pin (4,11) and the fictitious
(d) Find the shortest path from all path rectangle three corners

4.4 An ECO Problem: After Inserting New Obstacles

Roughly, there are three aspects of ECO issues: (1) resize the network, (2) location migration, (3) circuit changes. It is more difficult in circuit changes in particular of all mentioned above, since the non-use spaces must be found to be rerouted in our circuits. The new obstacles are inserted to change our original circuits, and the paths which are needed to be change are routed by using our strategies.

It wastes much time and resources to reroute all circuits, especially in only a little change or modifying many times by engineer. Rerouting all circuits several times will cost very much, therefore unnecessary routing work must be diminished to avoid rerouting all circuits when rerouting work needs to be implemented. It is considered that narrowing rerouting area such that rerouting in only partial area is implemented. However, still some unnecessary rerouting works are done while narrowing rerouting area is implemented. What we need to do first is to find out the influenced routing paths and to do necessary rerouting work of the changing paths.

In addition, whether the load of routing work is plenty or not, it takes much and unnecessary algorithm process time from the beginning to the end while rerouting work are implemented by original router. Therefore, we reserve the information constructed during routing process. The associated paths which need to be modified can be found quickly by applying the information while rerouting process. And then only necessary approaches must be done to find out the modified routing paths. The rerouting results are outputted at last. In this way, we can save plenty of unnecessary time and get rerouting results more quickly.

Table 1 The comparison of wirelength and the run time between Manhattan and our Xroute result.

	Pin	Obs.	Manhattan		Xroute		Comparison	
			time	wirelength	time	wirelength	time(%)	wirelength(%)
case1	60	20	0.078	130800	0.078	120705	0%	7.72%
case2	140	20	0.21	217150	0.203	215768	3.33%	0.64%
case3	200	20	0.343	249390	0.359	245280	-4.66%	1.65%
case4	200	600	7.125	375698	7.093	356647	0.45%	5.07%
case5	1000	200	9.078	574455	9.265	555387	-2.06%	3.32%
case6	400	1000	22.39	495111	22.593	471873	-0.91%	4.69%
case7	2000	200	24.375	740804	25.609	726347	-5.06%	1.95%

Table 2 The comparison of wirelength and the run time between [11] and our Xrouting result.

	Pin	Obs.	[11]		Xroute		Comparison	
			time	wirelength	time	wirelength	time(%)	wirelength(%)
case1	60	20	0.078	132395	0.078	120705	0%	8.83%
case2	140	20	0.203	218820	0.203	215768	0%	1.39%
case3	200	20	0.312	271285	0.359	245280	-15.06%	9.59%
case4	200	600	6.56	379657	7.093	356647	-8.13%	6.06%
case5	1000	200	8.34	602840	9.265	555387	-11.09%	7.87%
case6	400	1000	19.015	524782	22.593	471873	-18.82%	10.08%
case7	2000	200	23.64	789107	25.609	726347	-8.33%	7.95%

5 Experimental Result

The proposed algorithms were implemented by using C++ language on an AMD 1.84G machine with 768MB memory. We obtain 25 benchmark circuits from [11], and we randomly choose six cases to test our algorithm. There are five tables of the experiment data.

5.1 Comparison between Our Approach and Manhattan-Wiring Approach

In this subsection, we focus on the comparison between our approach and Manhattan-wiring approach. The Manhattan-wiring algorithm and our algorithm apply the same method to insert via, thus the structures of both are almost the same. The extreme difference of them is that the routing paths are implemented by different ways. The former uses the rectilinear path; and ours uses the X-Architecture path. Table 1 demonstrates that the X-Architecture has the better wirelength and almost the same run time.

5.2 Comparison between Our Approach and the Approach in [11]

In this subsection, we focus on the comparison between our approach and the approach in [11]. All pins are split up in [11]. First, all pins are projected to the same layer, and then they are divided into four groups according the x and y coordinate. The center of each group is found and the via is inserting to the point.

According to the via point, each group is divided into four regions. The routing of each region is implemented in order until all of them are implemented. Table 2 shows that wirelength of our approach is better but the run time is slightly worse than the algorithm in [11].

Table 3 The comparison of skew.

	Pin	Obs.	Skew			Comparison	
			Manhattan	[11]	Xroute	Manhattan(%)	[11](%)
case1	60	20	23190	18050	20824	10.2%	-15.37%
case2	140	20	33820	33820	31407	7.13%	7.13%
case3	200	20	29150	28400	27136	6.9%	4.45%
case4	200	600	34573	33919	33348	3.54%	1.68%
case5	1000	200	42710	51800	39582	7.32%	23.59%
case6	400	1000	39479	41856	37922	3.94%	9.4%
case7	2000	200	76141	74893	71816	5.68%	4.1%
Average						6.39%	5%

Table 4 The obstacle-avoiding routing result of rectangle obstacles and non-rectangle obstacles.

	Pin		Obstacles		Time	Wirelength
	original pin	inside pin	original obs.	non-rectangle obs.		
nrobcase	20	1	20	2	0.047	76095

5.3 The Skew Comparison

According to the two experiments above, we calculate the skew for comparison. Table 3 shows that skew of our approach is averagely better than other methods (the ratio of via = 200). Compared with Manhattan approach and the algorithm in [11], the experimental results show that our method improves the skew by 6.39% and 5%, respectively.

5.4 Non-Rectangle Obstacle Avoiding Routing

In tale 4, we give an example for the non-rectangular obstacle avoiding routing. Twenty rectangle obstacles, twenty-one pins, and two non-rectangle obstacles are included to test our algorithm (isosceles right triangles is used as the non-rectangle obstacles in our experiment). One pin is located exactly inside the routable fictitious rectangle region from the non-rectangle obstacles. The spanning graph of the inner pin is established and combined with other spanning graph. The shortest path is found at last. Table 4 shows the routing result include the rectangle and non-rectangle obstacles by our algorithm.

Table 5 The rerouting result of inserting a new obstacle.

	Pin	Obstacles		Time		Wirelength	
		obstacles	new obs.	all reroute	eco reroute	all reroute	eco reroute
ecocase1	20	20	1	0.031	0.01	77663	77130
ecocase2	2000	200	1	24.172	0.422	726862	726370

5.5 Rerouting for Obstacle Inserting

Two ECO routing examples are implemented, one case includes twenty pins and twenty obstacles, and the other one includes two thousand pins and two hundred obstacles. Suppose a new obstacle should be inserted into both cases respectively and we try to find the routing path. In table 5, we compare reroute whole circuit and our ECO approach separately. The result shows that our approach is efficient and effective.

6 Conclusion

In this paper, under the conditions of rectangle obstacles and non-rectangle obstacles, we use fewer vias and X-Architecture router by region to construct the multilayer routing trees. The main purpose is to obtain a routing tree of minimal wirelength and minimal delay. On the other hand, in order to solve one of the ECO (Engineering Change Order) problems, we keep the previous routing information to find the path, which needs to be modified, and thus the rerouting can be built quickly and efficiently. Compared with the traditional ECO method that reroutes all the nets again, we can save a lot of run time and resources. With additional consideration for the non-rectangle obstacles routing, our router is able to deal with the probable problems under different conditions. According to our experimental results, we can obtain the better wirelength and skew than other methods.

References:

[1] M. Berg, M. Krcveld, M. Overmars, and 0. Schwarzkapf. “Computational Geometry: Algorithms and Applications”. In *2nd Edition, Springer-Verlag*, 2000.

[2] C. F. Chang and Y. W. Chang. “X-Route: An X-Architecture Full-Chip Multilevel Router”. In *Proceedings IEEE International SOC Conference*, pages 229–232, 2007.

[3] S. P. Chang, H. H. Huang, Y. C. Lin, and T. M. Hsieh. “A Timing-Driven X-Architecture Router with Obstacles”. In *VLSI Design/CAD Symposium*, 2007.

[4] J. Cong, Jie Fang, and K. Y. Khoo. “An implicit connection graph maze routing algorithm for ECO routing”. In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 163–167, 1999.

[5] J. L. Ganley and J. P. Cohoon. “Routing a Multi-Terminal Critical Net: Steiner Tree Construction in Presence of Obstacles”. In *Proceedings International Symposium on Circuits and Systems*, pages 113–116, 1994.

[6] S. Hema, V. Suresh and P. Ramesh, “FPGA Implementation of Viterbi Decoder,” in *Proceedings of the 6th WSEAS Int. Conference on Electronics, Hardware, Wireless and Optical Communications*, 2007, pages 162-167.

[7] T. Y. Ho, C. F. Chang, Y. W. Chang, and S. J. Chen. “Multilevel Full-Chip Routing for the X-Based Architecture”. In *Proceedings IEEE/ACM Design Automation Conference*, pages 597–602, 2005.

[8] Y. Hu, T. Jing, X. Hong, Z. Feng, X. Hu, and G. Yan. “An-OARSMan: Obstacle-Avoiding Routing Tree Construction with Good Length Performance”. In *Proceedings ACM/IEEE Asia and South Pacific Design Automation Conference*, pages 7–12, 2005.

[9] H.H. Huang, H.Y. Huang, D.J. Huang, T.M. Hsieh, “Efficient Obstacle-avoiding Rectilinear Steiner Tree Construction Algorithms”, In *WSEAS Transactions on Circuits and Systems*, 2006, pages 1775 – 1782.

[10] H.H. Huang, H.Y. Huang, D.J. Huang, T.M. Hsieh, “An Efficient Rectilinear Steiner Tree Algorithm with Obstacles”, In *5th WSEAS*

International Conference on Circuits, Systems, Electronics, Control and Signal Processing (CSECS '06), 2006，pages 54-59.

[11] H. H. Huang, Y. C. Lin, H. Y. Huang, and T. M. Hsieh. "Partition-based Routing Tree Algorithm with Obstacles". In *Proceedings of IEEE International Symposium on Integrated Circuits*, pages 196–199, 2007.

[12] A. B. Kahng and G. Robins. "A new Class of Steiner Tree Heuristics with Good Performance". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 11, pages 893–902, 1992.

[13] C. Y. Lee. "An Algorithm for Connections and Its Application". In *IRE Transactions on Electronic Computer*, pages 346–365, 1961.

[14] Y. L. Li, J. Y. Li, and W. B. Chen. "An Efficient Tile-Based ECO Router Using Routing Graph Reduction and Enhanced Global Routing Flow". In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 345–358, 2006.

[15] S. K. Lim. "Physical design for 3D system on package". In *IEEE Design and Test of Computers*, volume 22, pages 532–539, 2005.

[16] K. Mikami and K. Tabuchi. "A Computer Program for Optimal Routing of Printed Circuit Conductors". In *Proceedings of IFIP Congress*, volume 2, pages 1475–1478, 1968.

[17] J. M. Minz, E. Wong, M. Pathak, and S.K. Lim. "Placement and Routing for 3-D System-On-Package Designs". In *IEEE Transactions on Components and Packaging Technologies*, volume 29, pages 644–657, 2006.

[18] J. R. Minz, E. Wang, and S. K. Lim. "Thermal and Crosstalk-Aware Physical Design for 3D System-On-Package". In *Proc. of IEEE Electrical Performance of Electronic Packaging*, pages 824–831, 2005.

[19] M. Pathak and S. K. Lim. "Thermal-aware Steiner Routing for 3D Stacked ICs". In *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pages 205–211, 2007.

[20] R. Ravichandran, J. Minz, M. Pathak, S. Easwar, and S. K. Lim. "Physical layout automation for system-on-packages". In *IEEE Electronic Components and Technology Conference*, 2004.

[21] Z. Shen, C. C. N. Chu, and Y. M. Li. "Efficient Rectilinear Steiner Tree Construction with Rectilinear Blockages". In *Proceedings IEEE International Conference on Computer Design*, pages 38–44, 2005.

[22] X.Y. Wang, L. Hedrich, "Hierarchical Symbolic Analysis of Analog Circuits Using Two-Port Networks," In *6th WSEAS International Conference on Circuits, Systems, Electronics, Control and Signal Processing (CSECS'07)*, 2007, pages 21-26.

[23] Y. Yang, Q. Zhu, T. Jing, and X.L Hong. "Rectilinear Steiner Minimal Tree among Obstacles". In *Proc. of IEEE ASIC 5th International Conference*, pages 348–351, 2003.