DraXRouter: Global Routing in X-Architecture with Dynamic Resource Assignment*

Zhen Cao[1], Tong Jing[1], Yu Hu[2], Yiyu Shi[2], Xianlong Hong[1], Xiaodong Hu[3], Guiying Yan[3]

[1] Computer Science & Technology Department
Tsinghua University
Beijing 100084, China
Phone: +86-10-62785564
Fax: +86-10-62781489
e-mail: caoz@mails.tsinghua.edu.cn

[2] Electrical Engineering Department
UCLA
Los Angeles, CA 90095, USA
Phone: (310) 267-5407
Fax: (310) 267-5407
e-mail: {hu, yshi}@ee.ucla.edu

[3] Institute of Applied Mathematics
Chinese Academy of Sciences
Beijing 100080, China
Phone: +86-10-62639192
Fax: +86-10-62574529
e-mail: {xdhu, yangy}@amss.ac.cn

Abstract – In recent years, the X-Architecture is introduced to obtain better performance for integrated circuit physical design. This paper reformulates the global routing problem in X-Architecture under the liquid routing model. Then, a dynamic resource assignment (Dra) method is presented to reduce potential vias. At last, a global router called DraXRouter, is designed, in which we adopt a dynamic-tabulist-based tree construction algorithm and a stochastic optimization strategy to gain high quality routing solution. Tested on ISPD'98 benchmarks, DraXRouter achieves better routing performance compared with two recent global routers.

I. Introduction

With advance in fabrication technology of integrated circuit (IC), the interconnect delay has become a significant factor affecting IC performance. The optimization capability of algorithms for interconnect performance optimization is limited since they were designed based on rectilinear interconnect architecture. Then, researchers begin to focus on other on-chip interconnect architectures to obtain better optimization results and higher performance. [4] indicates that non-Manhattan routing/interconnect optimization is now championed by the X-Architecture.

Compared with traditional Manhattan architecture, the X-Architecture reduces wire length by 20%, as well as via number by 30%. This results in a chip performance improvement of 10% and a power reduction of 20% [1]. However, the X-Architecture is not widespread today due to the lack of X-Architecture-based placement and routing algorithms.

In the traditional Manhattan architecture, some recent progresses focus on congestion reduction. [3] presented a global router, called labyrinth, which evaluates placement results in terms of congestion and wire length. By integrating amplified congestion estimates with the routing, [16] improved over-congestion and routing length. [12] presented an efficient congestion reduction algorithm for global routing based on search space traversing technology (SSTT), by which large circuits can be routed in a short running time while keeping good performance.

The routing problem in traditional architecture is well studied while there is not much literature focusing on the X-Architecture-based routing problem. [4] proposed a new global router to explore modern interconnect problems in Manhattan and non-Manhattan architectures, which indicated that non-Manhattan architectures require more vias than Manhattan architecture under preferred direction routing. [15] introduced a layer balance approach for congestion reduction in both Manhattan and non-Manhattan architectures. [2] presented a routing algorithm with high complexity under liquid routing to reduce both total wire length and the number of vias, while keeping high routability. [20] proposed the first multilevel routing framework for the X-Architecture. Compared with router in Manhattan Architecture, this router can reduce wire length by 18.7% and average delay by 8.8% with similar routing completion rates. Via number was not reduced.

Liquid routing model [1], [18] is an effective way to reduce vias in X-Architecture. [10] considers the liquid routing model in global routing phase, and studies on the routing resource estimation method for liquid routing model in X-Architecture, based on which a global routing algorithm is presented. Compared with labyrinth [3] and SSTT [12] in Manhattan architecture, this algorithm can reduce the total wire length and total overflow more than 10% and 80% on average, respectively.

To explore the more benefit can be obtained from the X-Architecture routing, this paper employs liquid routing model in global routing. Our work aims to find a more accurate and effective way to estimate and utilize the routing resources in global routing stage and provide a better guidance for liquid routing after global routing. The main contributions of this paper are as follows.

(1) We present a dynamic resource assignment (Dra) method to guide liquid routing in X-Architecture, in order to utilize the routing resources more reasonably under liquid routing model.

(2) We design a global router, called DraXRouter, in which we adopt a dynamic-tabulist-based (DTB) tree construction algorithm and the stochastic optimization strategy to gain better routing performance.

Tested on ISPD'98 benchmarks, our router, DraXRouter, has gained good routing performance. Compared with x-labyrinth[1], DraXRouter improves utilization of routing resource with a shorter wire length. At the same time, the

* This work was supported in part by the NSFC under Grant No.60373012 and the SRFDP of China under Grant No.20050003099.

[1] We designed labyrinth algorithm [7] in X-Architecture, called x-labyrinth, for more comparison with our DraXRouter.

0-7803-9451-8/06/$20.00 ©2006 IEEE.

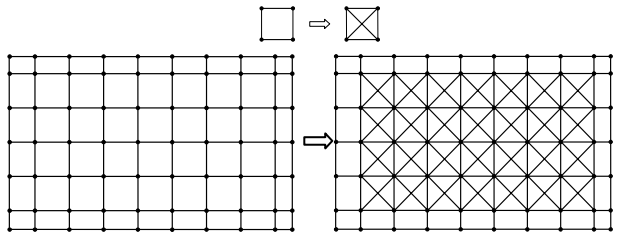
runtime of DraXRouter is 33% shorter. Compared with COCO algorithm [10], DraXRouter gains a better routing result with an acceptable runtime. Meanwhile, we compared the wire crossover in the global routing results produced by DraXRouter with the traditional resources assignment (RA) methods and that with the Dra method, respectively. The results indicated the Dra method can reduce the crossover number by 5.09% with even a shorter wire length.

The remainder of this paper is organized as follows. In Section II, we formulate the global routing problem in X-Architecture. Section III presents the Dra method. Section IV describes the details of the global routing algorithm and the implementation of our global router, DraXRouter. Experimental results are given in Section V. We conclude this paper in Section VI.

II. Preliminaries

A. GRG Generation in X-Architecture

In global routing problem people often partition the routing area into several global tiles or global routing grid (GRG) and map all physical pins into each tile, and then find a routing to connect the center of corresponding tiles. There exist several ways to partition routing area in non-Manhattan architectures. For X-Architecture, [8] presented the octagonal tiling. Since our global router was supposed to guide a gridless liquid detail routing, it's hard to use this GRG generation. As the placement tool in X-Architecture is not mature [1], we utilize the placement result of a traditional standard cell placer as the input of our global router. We adopt the GRG partition method in [10]. As Fig.1 shows, we add some diagonal edges to connect each two diagonal adjacent vertices.



(a) (b)

Fig. 1. GRG generation (a) GRG in Manhattan architecture, (b) GRG in X-Architecture.

B. Problem Formulation

Given c_e as the capacity of a GRG edge, d_e as the routing demand for edge *e*, the overflow of edge *e* is defined as follows.

$$overflow_e = \begin{cases} d_e - c_e, & if\ d_e > c_e \\ 0, & otherwise \end{cases}$$

The total overflow of the entire routing area is as follows.

$$tof = \sum_{e \in E} overflow_e \quad (1)$$

The total wire length can be calculated as follows.

$$twl = \sum_{n=1}^{N_{net}} length_{net_n} = \sum_{n=1}^{N_{net}} \sum_{t=1}^{N_{edge}} l(e_t) \quad (2)$$

Then, the global routing problem can be formulated as find routing solution for each edge *e*, such that minimize:

$$f(tof, twl) \quad (3)$$

This function considers both wire length and total overflow, will be introduced in detail in Section IV.B.

C. Design Flow Under Liquid Routing Model

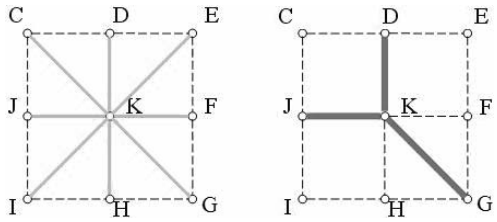
As routing with preferred direction in X-Architecture may cause an increase of via number, people adopt liquid routing [18], which utilize the gridless octilinear routing technology, to finish the detailed routing.

Our routing algorithm is designed to guide liquid detailed routing, so a layer assignment process should be employed to assign the intersectant segments into different layers.

III. Dynamic Resource Assignment

A. Capacity Upper Bound C_{eub}

Researches adopted liquid routing [1] to reduce via number in X-Architecture. However, in liquid routing model, the routing resource is not static. Routing in one edge will influence its adjacent edges. Considering the GRC (global routing cell) shown in Fig.2(a), K is the center of GRC CEGI, also a vertex in GRG. Routing in the direction KG will affect the routing resources of KF and KH. Routing in direction of KF will affect the routing resources of KE and KG.



(a) (b)

Fig.2. Routing resources interaction in liquid routing model (a) GRC in X-Architecture, (b) an extreme routing demand.

Fig.2(b) presents a case with an extreme routing demand. That is, the routing demand of KG is too much and the routing resources of KF and KH are all utilized by KG. Meanwhile, the routing resources of KE, KC, and KI are all utilized by KD and KJ (three blue segments in Fig.2(b)). In traditional static RA, such requirements can not be met due to the fixed pre-assignment. I.e. the pre-assignment resources for each edge can not be changed during the whole routing process.

In order to assign the routing resources more reasonably, we present a dynamic resource assignment (Dra) method to take on characters of liquid routing model, which assigns the routing resources in the routing process. In Dra method, a capacity upper bound C_{eub} (NOT a fixed pre-assignment capacity value) is given to each routing edge, which indicates

the potential routing resources utilized by this edge. First, the C_{eub} of vertical and horizontal edges are calculated based on wire widths. Then, the C_{eub} of diagonal edges is estimated based on an area distribution. In the routing process, the C_{eub} of an edge is influenced by both routing over this edge and routing over its adjacent edges. Then, the routing resources will be assigned dynamically in the routing process to take on characters of liquid routing model.

B. Calculation of C_{eub}

The capacities of vertical and horizontal edges are calculated based on wire widths in traditional method. Then, we take half of the routing area of vertical and horizontal edges as the routing area for diagonal edges. As Fig.3 shows, assume the capacity of vertical edges is 20. As the capacity is taken by half, 10 left for vertical edges. And the capacity of diagonal edges is $10\sqrt{2}$ as the width of track is not changed.

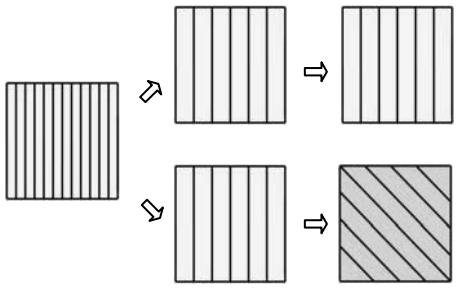


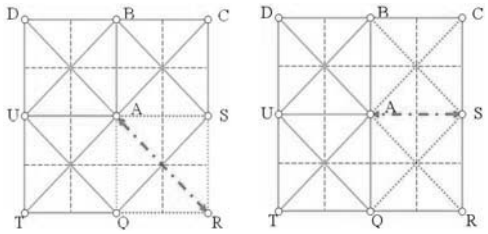
Fig.3. Calculation of capacity.

The C_{eub} of an edge indicates the potential routing resources utilized by this edge, then the initial C_{eub} should be twice of the capacity due to our calculation method. Meanwhile, Dra should guarantee the real resources utilized is the same as that of traditional method. Suppose the C_{eub} of a vertical edge is c_v and the C_{eub} of a horizontal edge is c_h, so the C_{eub} of a diagonal edge is $\sqrt{c_v^2 + c_h^2}$.

C. Dynamic Resource Assignment (Dra)

The assignment of Dra considers the influence of routing on one edge to its adjacent edges. See Fig.4(a) as a reference, if a net is routed over edge AR, the C_{eub} of AR will be cost one. As an influence of AR, the C_{eub} of AS and RQ will be taken a cost of $c_h / 4\sqrt{c_v^2 + c_h^2}$. And the C_{eub} of AQ and SR will be taken a cost of $c_v / 4\sqrt{c_v^2 + c_h^2}$. In Fig.4(b), if a net is routed over AS, the influence of AS to AR, SQ, SB, and AC will be $\sqrt{c_v^2 + c_h^2} / (4 * c_h)$. Similarly, the influence of AQ to AR, QS, AT, and QU will be $\sqrt{c_v^2 + c_h^2} / (4 * c_v)$.

The proportion of Dra method comes from the prove process as follows. Assume that the routing resources of a GRG are just utilized by all horizontal and vertical routing edges or all by diagonal edges. That is, c_v vertical edges and c_h horizontal edges or $2\sqrt{c_v^2 + c_h^2}$ diagonal edges. This indicates these routing edges share the same routing resources. Then, the proportion of interaction can be calculated. This proportion guarantees the real resources utilized are the same as the traditional methods.



(a) (b)

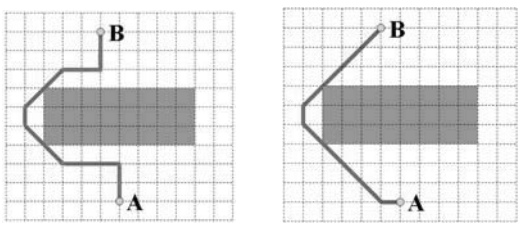
Fig.4. The Dra method in X-Architecture.

IV. DraXRouter Routing Algorithm

In this section, we design a new X-architecture-based global router, called DraXRouter. In DraXRouter, we construct initial trees firstly, and then rip-up and reroute congested nets to reduce congestion with several iterations. We first present our tree construction heuristic.

A. Octilinear Steiner Tree Construction Heuristic

Steiner minimal tree problem has been proved to be NP-complete in [19]. A random sub-trees growing heuristic (RSG) is introduced in [10] as a tree construction algorithm for global routing. This heuristic is efficient and can find Steiner points for the terminals to be connected. But it just searches the next one step of a growing point, which may cause an improper detour. As shown in Fig.5(a), the pins to be connected is A and B. The gray area indicates congestion. If the heuristic only searches one step, the detour will be started too late. The accurate detour is shown in Fig.5(b). In X-Architecture, routing can explore more directions than Manhattan cases, which indicates that we may get much worse results without a good control of the sub tree growing.



(a) (b)

Fig.5. Detour in X-Architecture.

We design a dynamic-tabulist-based algorithm (DTB) to solve this problem, in which every pin of the net is initially regarded as a growing point (GP). And each GP will grow toward another GP according to the others' tabulist. When two GPs meet, they will merge into a new GP. This process repeats until there is one GP left and the tree is constructed.

Algorithm 1 DTB Heuristic
Input: Graph *G* and terminal set *N*
Output: An octilinear Steiner minimal tree in *G*

```
1:  Put a GP on every terminal.
2:  while the number of GPs > 1 do
3:      t ← select a GP p according to estimate function (4), t is its current position;
4:      p grows from t;
5:      find the new current position v for p;
6:      if v ∈ tabulist of another GP p' then
7:          p_new ← merging p and p';
8:          Compute the location of P_new;
9:      else
10:         update the tabulist of p;
11:     end if
12: end while
13: return tree;
```

Fig.6. The pseudo-code of DTB.

When growing, the current position of a GP and its tabulist is selected dynamically to find out the best topology for the Steiner tree. The tabulist of a GP is changed in construction process, and the influence of this GP to others is changed too. All GPs interact until the tree is constructed. The pseudo-code of DTB is shown in Fig.6.

The estimate function is defined as follows.

$$f_p(v) = w(p,v) + dis(s,v) + g(p,v) \quad (4)$$

Where *w(p,v)* presents the congestion information of the tabulist of *p* from source to *v*, *dis(s,v)* presents the routing distance from source to *v*, *g(p,v)* is the heuristic function, it allows every GP to grow toward other GPs.

The construction process of an 11-pin net is shown in Fig.7. Every GP is given a color. The color of a wire is the same as color of the active GP on this wire. The gray area indicates congestion. The tabulist of the orange GP affects tabulist of the green GP in the routing process, as shown in Fig.7(b) and Fig.7(c). Final topology of the tree is shown in Fig.7(d), in which the best topology is obtained. This shows DTB algorithm has an advantage of avoiding congestion and finding proper Steiner point.

B. Stochastic Optimization Approaches

Our routing algorithm uses DTB to construct initial trees. Then, several iterations are required for the rip-up and reroute process to refine the solution. For every iteration, we first find out all nets that go over a congested edge and pick up the nets to be rerouted randomly according to an proportion *p* calculated in Eq.(6).

$$p = P * cnum / tof \quad (6)$$

where *cnum* is the number of congested nets, *tof* is the total overflow by Eq.(1), *P* is a constant.

Then, these nets are rerouted using DTB heuristic with the new routing demand estimation. The new *tof* and *twl* are calculated in Eq.(1) and (2). After the new routing solution of these nets is obtained, a judgment of whether to accept this routing solution is employed. This judgment is formulated as follows.

$$\begin{cases} d_{tof} \geq 0 \; and \; d_{twl} \geq 0 & p = 0 \\ d_{tof} < 0 \; and \; d_{twl} \leq 0 & p = 1 \\ d_{tof} < 0 \; and \; d_{twl} > 0 & p = e^{(d_{twl}/-d_{tof} - T)} \\ d_{tof} > 0 \; and \; d_{twl} < 0 & p = e^{(-d_{twl}/d_{tof} - T)} \end{cases}$$

where p is the probability of acceptance, T is defined as T = (N / K + 1) * D, N is the number of iterations, D is a constant, dtof and dtwf are the difference of the total overflow and total wire length between the previous iteration and the current iteration. If this solution is accepted, the new topologies will be stored and the new routing demand will be utilized by the next iteration. This judgment can avoid trapping into a local minimum.

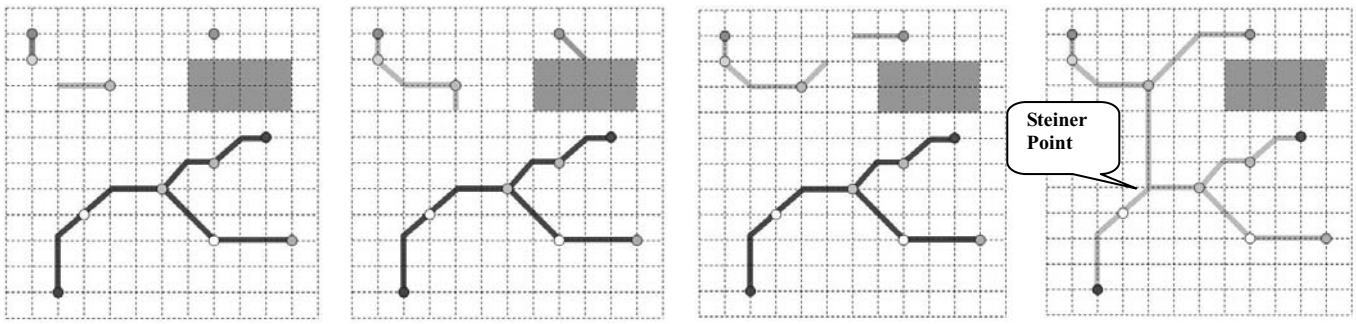
V. Experimental Results

A. Experiments Setup

We test our global router with the ISPD'98 benchmarks. TABLE I shows the characteristics of all benchmarks.

As there are few global routers in X-Architecture, we studied the routing algorithm of labyrinth [3] and implemented it in the X-Architecture, called x-labyrinth, for comparison.

To demonstrate the optimization capability of our router, we compare our router with x-labyrinth and COCO [10]. For a fare comparison, we assign the same routing resources to all the three routers. For these three routers all have strong capabilities of congestion reduction, the experimental results of initial routing resource provided by the benchmarks show



(a) (b) (c) (d)

Fig.7. Interaction between dynamic tabulists in tree construction process.

that there exist no congestion, then it is hard to compare optimization capability of these routers, so we shrink the routing resources until overflow appears. The total overflow (*tof*), total wire length (*twl*) and runtime (CPU) of the routing results are compared and shown in TABLE II to TABLE V. Experiments in this paper are performed in 1.6GHz CPU / 680MB memory Linux.

TABLE I
Benchmark Data

Circuits	Net#	Grids	Circuits	Net#	Grids
ibm01	13k	64×64	ibm06	34k	128×64
ibm02	19k	80×64	ibm07	46k	192×64
ibm03	26k	80×64	ibm08	49k	192×64
ibm04	31k	96×64	ibm09	59k	256×64
ibm05	30k	128×64	ibm10	66k	256×64

To show the effectiveness of Dra method, we run our router under the static RA method [10] and Dra method, respectively. After the running, the total crossed segment numbers of both strategies are compared. The number of the crossovers indicates the number of potential vias, so the reduction of crossover brings reduction of via number in the gridless liquid detailed routing.

B. Wire length and Congestion Comparisons

The comparison of test results is shown in TABLE II to TABLE V. Compared with x-labyrinth, our algorithm can achieve 100% completion for global routing for almost all circuits (8 out of 10 testcases). On average, DraXRouter reduces total overflow by 99.49%, total wire length by 3.21%. Compared with COCO, our algorithm reduces total overflow by 99.93% and total wire length by 18.15% with an acceptable runtime.

TABLE II
Comparison between DraXrouter & x-laby on *twl*

Circuits	DraX	x-laby	Imp.(%)
ibm01	61011.9	65989.6	7.54
ibm02	168908	169388	0.28
ibm03	148970	163081	8.65
ibm04	163009	174081	6.36
ibm05	408902	421328	2.95
ibm06	275598	280265	1.67
ibm07	349383	355865	1.82
ibm08	389485	394140	1.18
ibm09	412195	416077	0.93
ibm10	565999	570143	0.73
Average	——	——	3.21

C. Crossover Number Reduction

In liquid routing, [2] indicates that any two crossed segments introduce a least one via. To show the effectiveness of the Dra method for via reduction in liquid routing, we compare the number crossed segments in the routing results produced by our router with and without the Dra Method. The crossed segments number is the sum of the crossover

TABLE III
Comparison between DraXrouter & x-laby on *tof* and Running Time

Circuits	DraX		x-labyrinth		*Tof*	CPU
	Tof	CPU(s)	*Tof*	CPU(s)	Imp.(%)	Imp.(%)
ibm01	4	36	87	89	95.40	59.55
ibm02	0	78	185	119	100.00	34.45
ibm03	1	73	217	148	99.54	50.68
ibm04	0	137	199	179	100.00	23.46
ibm05	0	438	109	587	100.00	25.38
ibm06	0	331	63	410	100.00	19.27
ibm07	0	349	80	548	100.00	36.31
ibm08	0	429	144	701	100.00	38.80
ibm09	0	734	33	987	100.00	25.63
ibm10	0	1132	96	1309	100.00	13.52
Average	——	——	—	——	99.49	32.71

TABLE IV
Comparison between DraXrouter & COCO on *twl*

Circuits	DraX	COCO	Imp.(%)
ibm01	61011.9	74256.8	17.84
ibm02	168908	201484	16.17
ibm03	148970	182767	18.49
ibm04	163009	197607	17.51
ibm05	408902	502694	18.66
ibm06	275598	329378	16.33
ibm07	349383	434956	19.67
ibm08	389485	475549	18.10
ibm09	412195	510459	19.25
ibm10	565999	702568	19.44
Average	——	——	18.15

TABLE V
Comparison between DraXrouter & COCO on *tof* and Running Time

Circuits	DraX		COCO		*Tof* Imp.(%)
	Tof	CPU(*s*)	*Tof*	CPU(*s*)	
ibm01	4	36	786	18	99.49
ibm02	0	78	299	46	100
ibm03	1	73	494	43	99.79
ibm04	0	137	1022	47	100
ibm05	0	438	900	236	100
ibm06	0	331	748	176	100
ibm07	0	349	503	159	100
ibm08	0	429	425	247	100
ibm09	0	734	1043	312	100
ibm10	0	1132	1183	350	100
Average	——	——	——	——	99.93

between nets. [2] adopts a similar method to calculate the crossover number but it takes area as a reference because it is considered in detailed routing. Obviously, the crossover number gives a good prediction for the number of vias produced by liquid routing.

The result is shown in TABLE VI, in which the router with the Dra method produces 5% lesser the crossed segments than the one with the traditional static RA method, and the wire length is even shorter. It indicates that the Dra method utilizes the routing resources more reasonably, so it provides a better guidance for liquid routing. Note that, both RA methods assign the same resource in the edges of a global routing graph to keep a fare comparison.

TABLE VI
Comparison between DraXrouter with old RA Method & DraXrouter with Dra Method

Circuits	Old		Dra		Cross
	Twl	Cross#	*Twl*	Cross#	Imp.(%)
ibm01	59804	234002	58031	223526	4.48
ibm02	162486	1521016	157341	1436766	5.54
ibm03	138126	1078131	134866	1012122	6.12
ibm04	156425	1081236	153320	1053306	2.58
ibm05	383450	5152323	381340	5041213	2.16
ibm06	268433	2289791	259542	2096781	8.43
ibm07	343957	2692672	337639	2541374	5.62
ibm08	382818	3213644	374226	3051586	5.04
ibm09	396096	2372184	385792	2215325	6.61
ibm10	548445	5377452	537612	5146802	4.29
Average	——	——	——	——	5.09

VI. Conclusions and Future Work

This paper reformulates the global routing problem in X-Architecture with the liquid routing model. A dynamic re-source assignment (Dra) method is presented to reduce potential vias. Based on our formulation, a global router, DraXRouter, is designed. In this router, we adopt a dynamic-tabulist-based tree construction algorithm and the stochastic optimization strategy to obtain better routing performance. Tested on ISPD'98 benchmarks, DraXRouter increases the routability substantially compared with two recent global routers, as well as reduces total wire length. The comparison with the traditional resource assignment method shows the Dra method can be a better guidance for liquid routing to reduce vias in detailed routing.

As future work, we will integrate the function considering timing and coupling issues into our router.

References

[1] Steven L Teig. The x architecture: Not your father's diagonal wiring. In Proceedings of the international work-shop on System-level interconnect prediction, pages 33–37, San Diego, California, USA., 2002.

[2] Martin Paluszewski, Pawel Winter, Martin Zachariasen. A New Paradigm for General Architecture Routing. GLSVLSI '04, April 26–28, 2004, Boston, Massachusetts, USA.

[3] Labyrinth. http://www.ece.ucsb.edu/~kastner/labyrinth/

[4] Cheng-Kok Koh and P. H. Madden. Manhattan or non-Manhattan? a study of alternative VLSI routing architectures. In Proceedings of the 10th ACM GLSVLSI, pages 47–52, Chicago, IL, USA, 2000.

[5] Qi Zhu, Hai Zhou, Tong Jing, Xianlong Hong, Yang Yang. Spanning Graph Based Non-Rectilinear Steiner Tree Algorithms. IEEE Trans. on CAD. 2005, 24(7).

[6] Majid Sarrafzadeh, C. K. Wong, Hierarchical Steiner Tree Construction in Uniform Orientations. IEEE Trans. on CAD, 11(9): pp.1095-1103, 1992.

[7] H. Chen, F Zhou and C. K. Cheng. The Y-architecture: yet another on-chip interconnect solution. In Proceedings of the ASP-DAC, pages 840-846, Kitakyushu, Japan, 2003.

[8] Yu Hu, Tong Jing, Xianlong Hong, Zhe Feng, Xiaodong Hu, Guiying Yan. An Efficient Rectilinear Steiner Minimum Tree Algorithm Based on Ant Colony Optimization. In: Proceedings of IEEE ICCCAS, 2004, Chengdu, China, 1276-1280.

[9] T. Mitsuhashi, K. Someha: Performance-oriented layout design, pervasive use of diagonal interconnects reduces wire-length. Design Wave Magazine (2001) 59–64

[10] Yu Hu, Tong Jing, Xianlong Hong, Xiaodong Hu, and Guiying Yan. A Routing Paradigm with Novel Resources Estimation and Routability Models for X-Architecture Based Physical Design. SAMOS V, Samos, Greece, 2005.

[11] S. Mand and W. C. K. An Introduction to VLSI Physical Design. McGraw Hill, USA, 1996.

[12] T. Jing, X. Hong, H. Bao, J. Xu, and J. Gu. SSTT: Efficient local search for GSI global routing. Journal of Compute Science and Technology, 18(5):632–639, September 2003.

[13] H. Chen, B. Yao, F. Zhou and C. K. Cheng, "Physical Planning of On-Chip Interconnect Architecture, In Proceedings of IEEE Int. Conference on Computer Design, pp. 30-35, Sept. 2002.

[14] The x initiative. http://www.xinitiative.org.

[15] A. R. Agnihotri and P. H. Madden. Congestion reduction in traditional and new routing architectures. In Proceedings of the 13th ACM GLSVLSI, pages 28–29, Washington, DC, USA., 2003.

[16] R. T. Hadsell and P. H. Madden. Improved global routing through congestion estimation. In Proceedings of the 40th conference on Design automation, Anaheim, CA, USA, 2003.

[17] H. Chen, B. Yao, F. Zhou and C. K. Cheng. Physical Planning of On-Chip Interconnect Architecture, In Proceedings of IEEE Int. Conference on Computer Design, pp.30-35, 2002.

[18] Takashi Mitsuhashi and Kenji Someha. Performance- Oriented Layout Design, Pervasive Use of Diagonal Interconnects Reduces Wire- Length, Design Wave Magazine, pages 59-64, September, 2001.

[19] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. SIAM Journal on Applied Mathematics, 32(4):835–859, 1977.

[20] Tsung-Yi Ho, Chen-Feng Chang, Yao-Wen Chang and Sao-Jie Chen. Multilevel Full-Chip Routing for the X-Based Architecture. DAC 2005, pages597-602, June 13–17, 2005, Anaheim, CA, USA.