Multilevel Full-Chip Routing for the X-Based Architecture

Tsung-Yi Ho[1], *Chen-Fong Chang*[1], *Yao-Wen Chang*[2*], *and Sao-Jie Chen*[2†]
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan[1]
Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan[2]

ABSTRACT

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. The X-architecture presents a new way of orienting a chip's microscopic interconnect wires with the pervasive use of diagonal routes. It can reduce the wirelength and via count, and thus improve performance and routability. Furthermore, the continuous increase of the problem size of IC routing is also a great challenge to existing routing algorithms. In this paper, we present the first multilevel framework for full-chip routing using the X-architecture. To take full advantage of the X-architecture, we explore the optimal routing for three-terminal nets on the X-architecture and develop a general X-Steiner tree algorithm based on the delaunay triangulation approach for the X-architecture. The multilevel routing framework adopts a two-stage technique of coarsening followed by uncoarsening, with a trapezoid-shaped track assignment embedded between the two stages to assign long, straight diagonal segments for wirelength reduction. Compared with the state-of-the-art multilevel routing for the Manhattan architecture, experimental results show that our approach reduced wirelength by 18.7% and average delay by 8.8% with similar routing completion rates and via counts.

Categories and Subject Descriptors

B.7.2 [Integrated Circuits]: Design Aids - Layout, Place and Route

General Terms

Algorithms, Designs

Keywords

Physical design, routing, multilevel optimization, X-architecture

1. INTRODUCTION

As integrated circuit geometries keep shrinking, interconnect delay has become the dominant factor in determining circuit performance. To minimize interconnect delay, two key IC technologies have been introduced: (1) copper and low-k dielectrics have replaced aluminum (as of the 180-nm and 130-nm nodes), reducing both resistance and capacitance, and (2) the ICs have been adapted to a new interconnect architecture, called the *X-architecture*, to shorten interconnect length and thus circuit delay.

The traditional Manhattan architecture has its obvious advantages of easier design (placement, routing, etc), but it adds significant and needless wirelength over the Euclidean optimum. As reported in [22], the average Manhattan wirelength is significantly longer than the average Euclidean distance. As shown in [1, 21, 23], the X-architecture's pervasive uses of diagonal routing can reduce wirelength and via count. In addition, the wirelength and via count reduction make the routing problem easier to solve, resulting in faster timing closure. These benefits contribute toward an increased probability of first-silicon success.

The most prevailing consortium that advocates routing at 45-degree increments is the X-initiative [1]. While lithographic considerations can impinge on the use of arbitrary angles for wiring, the use of 45-degree wires is fully supported by nearly all current manufacturing technologies. Recently, Toshiba and Cadence have launched the industry's first commercial system-on-chip (SoC) devices built on the innovative X-architecture design. Toshiba says that the TC90400XBG digital-media application processor is approximately 11% faster than comparable Manhattan-layout embedded chips in its product line. Thus, the X-architecture, the first production-worthy approach to the pervasive use of diagonal interconnect, shows promise to reduce the total interconnect while simultaneously improving the chip performance, power and cost.

Routing complexity is also an important problem for modern routers. To cope with the increasing complexity, researchers have proposed multilevel approaches to handle the problem [8, 9, 14, 15, 20]. The multilevel framework has attracted much attention in the literature recently [10]. It employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, routing tiles, etc.) based on a predefined cost metric, until the number of components being considered falls below a certain threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components and refines the solution by using a combinatorial optimization technique (e.g., simulated annealing, local refinement, etc). The multilevel framework has been successfully applied to partitioning, floorplanning, placement and routing in VLSI physical design.

Figure 1 shows our multilevel framework for the X-architecture. To take full advantage of the X-architecture, we explore the optimal routing for three-terminal nets on the X-architecture and develop a general X-Steiner tree (XST) algorithm based on the delaunay triangulation approach for the X-architecture. Given a netlist, we first run the XST algorithm to construct the topology for each net. We then decompose each net into 2-pin connections, with each connection corresponding to an edge of the XST. Our multilevel framework starts from coarsening the finest tiles of the lowest level. At each level, pattern routing for the X-architecture is used for routability-driven global routing. After the coarsening stage, we perform a trapezoid-shaped track assignment for diagonal segments. Most long, straight diagonal segments get track-assigned, and thus we can get lots of run-time improvement—the track-assignment process not only takes less computation time than detailed maze routing but also, only short segments (segments in lower levels) are delegated to the detailed router. In the uncoarsening stage, the unroutable nets are re-tried by point-to-path maze routing, rip-up and re-route to refine the routing solution level by level. Compared with the state-of-the-art multilevel routing [14] for the Manhattan architecture, experimental results show that our approach reduced wirelength by 18.7% and average delay by 8.8% with similar routing completion rates and via counts. The results show the promise of our approach.

It should be noted that the 18.7% improvement in wirelength is significantly better than the 11% improvement obtained by Toshiba's tool for routing its TC90400XBG digital-media application processor, as mentioned earlier. The difference also implicitly reveals the effectiveness of our multilevel routing.

The rest of this paper is organized as follows. Section 2 presents the routing model for the multilevel routing framework. Section 3 presents

* Yao-Wen Chang's work was partially supported by National Science Council of Taiwan under Grant No's. NSC 93-2215-E-002-009, NSC 93-2215-E-002-029, and NSC 93-2752-E-002-008-PAE.

† Sao-Jie Chen's work was partially supported by the National Science Council of Taiwan under Grant No. NSC 92-2218-E-002-032.

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

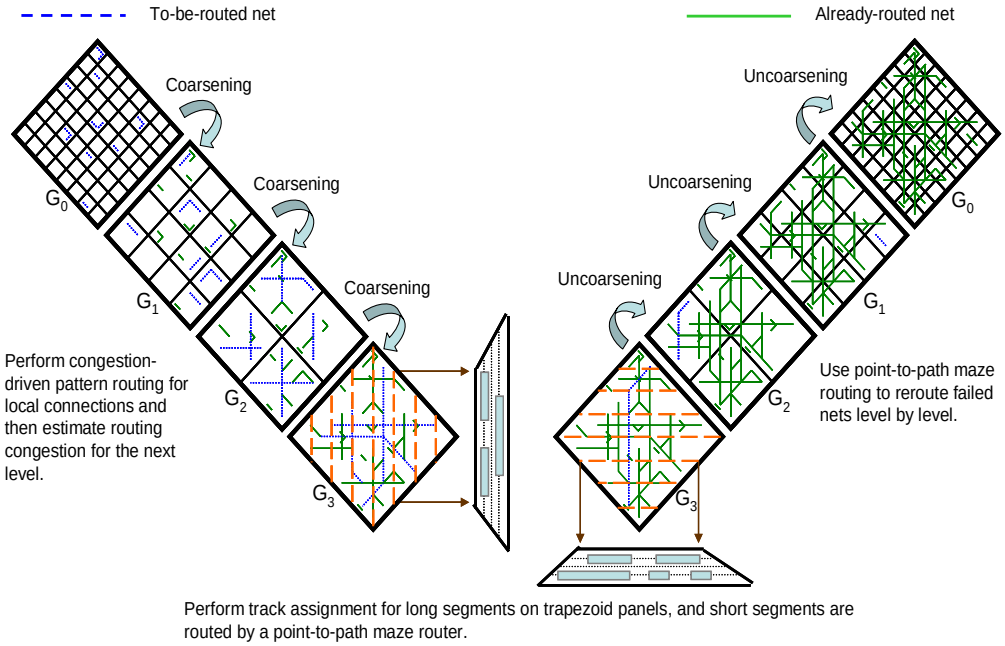

Figure 1: The multilevel framework flow

our novel multilevel routing framework for the X-architecture. Experimental results are shown in Section 4. In Section 5, we summarize our contributions and suggest future directions for research.

2. PRELIMINARIES

2.1 Routing Model

Routing in modern IC's is a very complex process, so we cannot easily obtain solutions directly. Our routing algorithm is based on a graph-search technique guided by the congestion information associated with routing regions and topologies, which assigns higher costs to nets passing through congested areas to balance the net distribution among routing regions. In this paper, we consider the four-layer routing cases for experiments on the X-architecture. In these cases, the first two layers are routed in the preferred direction H and V. Layers 3 and 4 are routed at eight compass directions (which is called *liquid routing*) to reduce the number of vias [1, 23].

Before we can apply the graph-search technique to multilevel routing, we first need to model the routing resource as a graph whose topology can represent the chip structure. Figure 2 illustrates the graph modeling. For the modeling, we first partition a chip into an array of rectangular subregions, each of which may accommodate tens of routing tracks in each dimension. These subregions are usually called *global cells* (*GCs*). A node in the graph represents a *GC* in the chip, and an edge denotes the boundary between two adjacent *GCs*. Then we add some diagonal edges to connect each two diagonal adjacent nodes to obtain the multilevel routing graph G_0 for the X-architecture. Each edge is assigned a capacity according to the physical area or the number of tracks of a tile. A global router finds *GC*-to-*GC* paths for all nets on G_0 to guide the detailed router. The goal of global routing is to route as many nets as possible while meeting the capacity constraint of each edge and any other constraint, that is specified.

2.2 Multilevel Routing Model

As illustrated in Figure 1, G_0 corresponds to the routing graph of the level 0 of the multilevel coarsening stage. At each level k, our global router just finds routing paths for the *local nets* (or *local 2-pin connections*) (those nets [connections] that entirely sit inside GC_{k+1}). After the global routing is performed, we merge four GC_k of G_k into a larger GC_{k+1} and at the same time perform resource estimation for use at level $k+1$. Coarsening continues until the number of *GCs* at a level, say the k-th level, is below a threshold. After finishing coarsening, a trapezoid-shaped track assignment is performed to assign the longer, straight diagonal segments to underlying routing resources. The uncoarsening stage task is to refine the routing solution of the unassigned segments that belong to level k where both pins are located in GC_{k+1}.

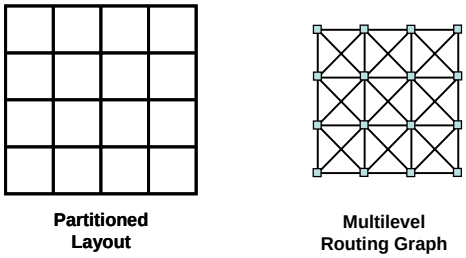

Figure 2: Routing graph.

During uncoarsening, the unroutable nets are directed to perform by point-to-path maze routing or rip-up and re-route, to refine the routing solution. Then we proceed to the next level (level $k-1$) of uncoarsening by expanding each GC_k to four finer GC_{k-1}. The process continues until we go back to level 0 when the final routing solution is obtained.

3. MULTILEVEL X ROUTING FRAMEWORK

Our multilevel routing algorithm is inspired by the work [14]. In the coarsening stage, a fast congestion-driven pattern routing is used for global routing, level by level. After the coarsening stage, we perform a trapezoid-shaped track assignment for diagonal segments. Most longer, straight diagonal segments get track-assigned, and thus can get lots of run-time improvement—the track-assignment process not only takes less computation time than detailed maze routing but also, only short segments (segments in level 0 and level 1) are delegated to the detailed router. In the uncoarsening stage, the unroutable nets are re-tried by point-to-path maze routing, rip-up and re-route to refine the routing solution level by level.

3.1 X-Architecture Steiner Tree Construction

The Steiner minimal tree problem has been proven to be NP-hard in [12]. In recent years, people have paid more attention to the algorithms for λ-geometry Steiner minimal tree problem. Coulston presented an exact algorithm for constructing exact octagonal Steiner minimal trees (OSMT) [11]. Kahng et al. proposed a highly scalable algorithm for both rectilinear and octilinear Steiner trees [16]. The most recent progress is Zhu's octilinear Steiner tree construction based on spanning graphs [24]. In this paper, we propose an X-Steiner tree algorithm based on the delaunay triangulation approach. By the optimal routing of each three-terminal net, we can extend the idea to construct our X-Steiner tree in $O(n \lg n)$ time.

3.1.1 Three-Terminal Net Routing Based on X-Architecture

Without loss of generality, given a two-terminal net $\delta = (1, 2)$, its optimal routing solution is one of the path around the parallelogram formed by δ (see Figure 3). Also, given any three-terminal net $\Gamma = (1, 2, 3)$, if terminal 3 is located in the merged region of the two bounding boxes of the other two terminals (see Figure 4), the optimal routing solution is the octilinear minimum spanning tree (OMST) of Γ inside this region.

Figure 3: Optimal routing of a two-terminal net.

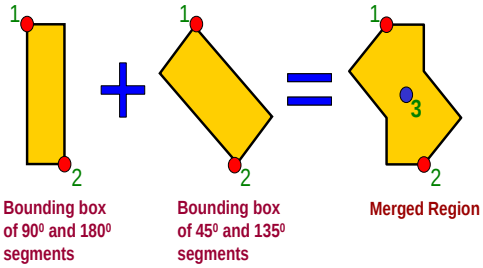

Figure 4: Merged region of two bounding-boxes.

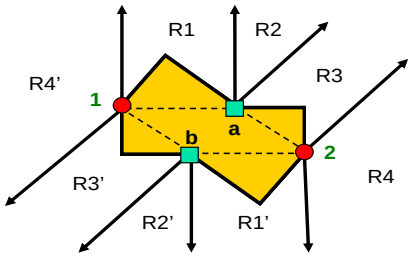

Figure 5: Octal regions of a two-terminal net.

LEMMA 1. *The optimal routing solution of a three-terminal net, of which one terminal is located in the merged region of the other two terminals, is the OMST of it.*

But if terminal 3 is not in the merged region, we can still find the optimal solution by connecting it to the nearest point of the previously-formed two-terminal net. For example, given a two-terminal net $\delta = (1, 2)$, the plane can be divided into eight octal regions as shown in Figure 5. If terminal 3 is located in R2 (R2'), we connect it to the Steiner point a (b), which is the apex of the parallelogram formed by terminals 1 and 2, to obtain the optimal routing solution (see Figure 5). If terminal 3 is located in R4 (R4'), we can take terminal 2 (1) as the internal terminal inside the merged region of terminal 1 (2) and 3, and the optimal routing solution can be obtained by Theorem 1. But if terminal 3 is located in R1 (R1', R3, or R3'), we divide the region into R1a and R1b by the vertical line D passing through the point V (see Figure 6 (a)). Without loss of generality, if terminal 3 is located in R1a, we can take terminal 1 as the third terminal to connect it to the Steiner point S of the two-terminal net formed by terminals 2 and 3 to obtain the optimal routing solution (see Figure 6 (b)). If the connected edge is perpendicular to the two-terminal net, we will refine the solution to a shorter wirelength. As shown in Figure 7, the refinement process will change the T-shaped portion (with length equal to $3\sqrt{2}$) of the net shown in the center part of Figure 7(a) to the L-shaped one (with length equal to 4) shown in Figure 7(b) to reduce the total wirelength.

The algorithm for three-terminal net routing on the X-architecture, called X3TR, is summarized in Figure 8. Since the numbers of terminals and wires being considered are both constant, we have the following theorem:

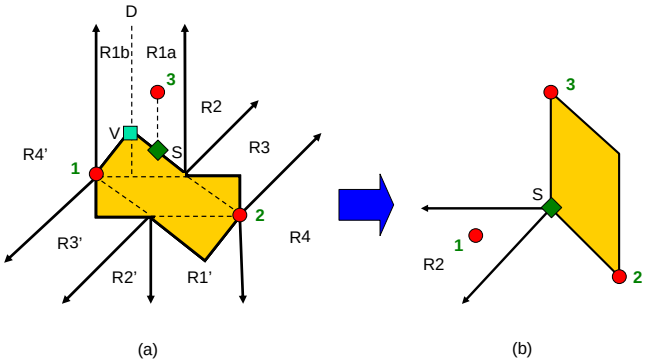

Figure 6: (a) If terminal 3 is located in region R1a, the optimal Steiner point will be S. (b) Terminal 1 is in "region 2" formed by terminal 2 and 3.

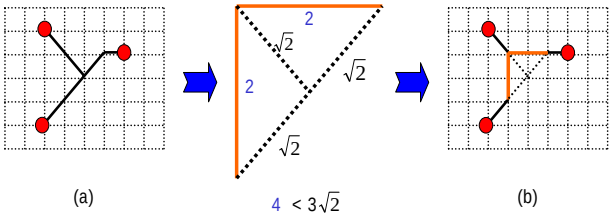

Figure 7: A line is perpendicular to another one, and refinement will result in the optimal solution.

THEOREM 1. *The X3TR algorithm finds the optimal routing of the minimum wirelength for a three-terminal net on the X-architecture in constant time.*

```
Algorithm : X3TR (3-Terminal Net Routing on X-Architecture)
Input     : A three-terminal net Γ = (α, β, γ);
Output    : The optimal routing tree T_o
begin
1  if (ThirdInsideMergedRegion(Γ)==True)
2     T_o = OMST(Γ);
3  else
4     OutsidePT = FindOutsidePT(Γ);
5     if (OctalRegion(OutsidePT)==R4 || R4')
6        T_o = OMST(Γ);
7     else if (OctalRegion(OutsidePT)==R2 || R2')
8        SteinerPT = ApexOfParallelogram(OutsidePT);
9        T_o = OMST(Γ, SteinerPT);
10    else if (OctalRegion(OutsidePT)==R1 || R1' || R3 || R3')
11       SteinerPT = VerticalOfParallelogram(OutsidePT);
12       T_o = OMST(Γ, SteinerPT);
end
```

Figure 8: Algorithm for three-terminal net routing based on X-architecture.

3.1.2 X-Steiner Tree Algorithm Based On Delaunay Triangulation

Since the optimal routing solution for each three-terminal net can be found easily, we use the delaunay triangulation approach [3] to divide all terminals into groups of three-terminal nets (see Figure 9 (a)). After that, we compute the optimal wirelength of all three-terminal nets, and sort them by their wirelength (see Figure 9 (b)). Further, we iteratively pick up a group of three-terminal nets with the minimal wirelength, then route and merge them to the X-Architecture Steiner Tree (XST) until it is constructed.

The time complexity for building the delaunay triangulation is $O(n \lg n)$, where n is the number of terminals [3]. And computing the optimal wire length of all three-terminal nets and sorting also take $O(n \lg n)$ time. Thus, the total time complexity for the XST construction is $O(n \lg n)$ time.

THEOREM 2. *The XST construction runs in $O(n \lg n)$ time, where n is the number of terminals.*

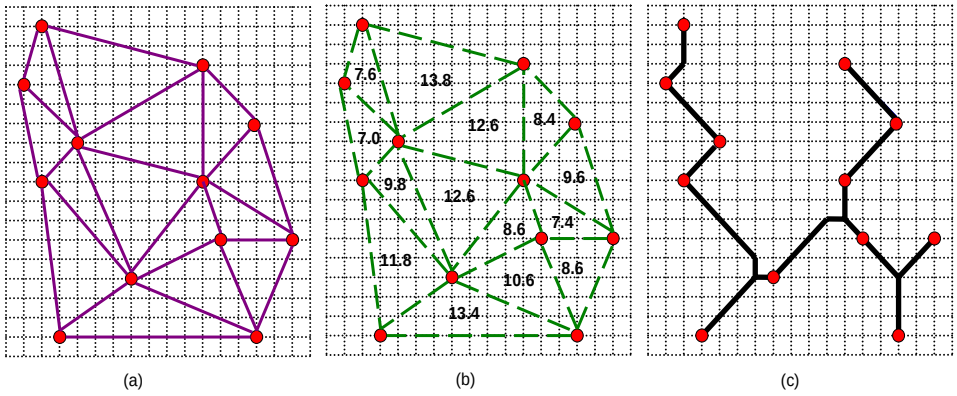

Figure 9: **(a) Delaunay triangulation of terminals (b) Optimal wirelength of each triangle. (c) XST.**

Algorithm : X-Architecture Steiner Tree
Input : Delaunay Triangulation DT of terminal set N
Output : X-Architecture Steiner Tree of DT
begin
1 For each terminal v in N
2 Set each v as a new subtree;
3 For each triangle T in DT
4 Compute the optimal wirelength of T;
5 Sort the wirelength of each T in DT in increasing order;
6 **while** (the number of subtree > 1) **do**
7 Route the triangle with minimal wirelength;
8 Merge the three subtrees to a new subtree;
9 Refine the routing result if needed;
end

Figure 10: **X-Architecture Steiner Tree Algorithm.**

The algorithm for building XST is stated in Figure 10, and the result of Figure 9 (a) is shown in Figure 9 (c).

3.2 Routability-Driven Pattern Routing

Given a netlist, we first run the XST algorithm to construct the topology for each net, and then decompose each net into 2-pin connections, with each connection corresponding to an edge of the XST. Our multilevel framework starts from coarsening the finest tiles of level 0. At each level, tiles are processed one by one, and only local nets (connections) are routed. The global routing which is based on the approach used in the pattern router [17] for the X-architecture, first routes local nets (connections) on the tiles of level 0. Figure 11 illustrates the pattern routing for the X-architecture. Let the multilevel routing graph $G_i = (V_i, E_i)$. We define $R_e = \{\ e \in E_i \mid e$ is the edge chosen to be routed$\}$. Then the cost of routing R_e is defined as:

$$cost(R_e) = \sum_{e \in E} c_e, \tag{1}$$

where c_e is the congestion of edge e and is defined by

$$c_e = 1/2^{(p_e - d_e)},$$

where p_e and d_e are the capacity and density associated with e, respectively.

Pattern routing uses a 1-bend or a 2-bend route to make the connection, whichever gives the shortest path length between two points. The wirelength is minimum, and thus we do not include it in the cost function at this stage. We measure the routing congestion based on the commonly used channel density. The channel density associated with an edge of a multilevel graph is updated level-by-level for fast resource estimation.

Our global router first tries 1-bend pattern routing. If those routing fails, we try 2-bend pattern routing. This can be considered as a simple version of rip-up and re-route. If both pattern routes fail, we give up routing the connection, and an overflow occurs. We refer to a *failed net* (*failed connection*) as one causes an overflow. The failed nets (connections) will be reconsidered (refined) at the uncoarsening stage. By this, we can efficiently obtain a good initial solution for the subsequent track assignment since pattern routing enjoys very low time complexity and uses fewer resources due to its simple 1-bend and 2-bend routing patterns.

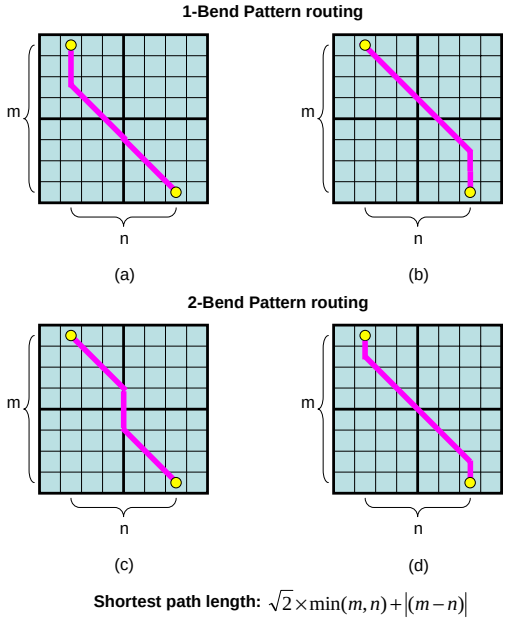

Figure 11: **Routing patterns for the X-Architecture.**

3.3 Trapezoid-Shaped Track Assignment

Due to the lithography issues of nanometer technology, the high via count is more likely to reduce yield. Reducing the number of vias is one of the key challenges for today's routers. In Figure 12, we show the difference of a two-pin connection between the Manhattan and the X-architecture. In this example, the total wirelength of the X-architecture is less than that of the Manhattan architecture ($1 + 2\sqrt{2} \approx 3.828 < 5$). But the via count of the X-architecture is larger than that of the Manhattan architecture ($3 > 1$). Therefore, if the wirelength of the two-pin net is short, the delay caused by via increase may offset the gains in the reduction of wirelength[23].

To overcome the drawback of via increase and fully utilize the benefit of wirelength reduction of the X-architecture, we assign only the long diagonal segments to tracks for better delay reduction.

In the gridded environment, each grid is λ apart from its immediate neighbors, where λ is the minimum spacing requirement dictated by the physical design rules. For the Manhattan architecture, this constitutes a perfect environment because there is at least λ distance between every gridpoint. But for the X-architecture, this commonly used grid-based model has a drawback: as shown in Figure 13(a), if a gridpoint has a 45(135)-degree wire passing through, topological design rules of the minimum spacing requirement dictates that the adjacent gridpoints cannot be used for routing ($\sqrt{2}\lambda/2 < \lambda$).

To overcome this drawback, we shift the aligned tracks to the *virtual tracks* for meeting the design rules (see Figure 13(b)). Although the virtual tracks are not aligned on the grids, we can use short wrong-way

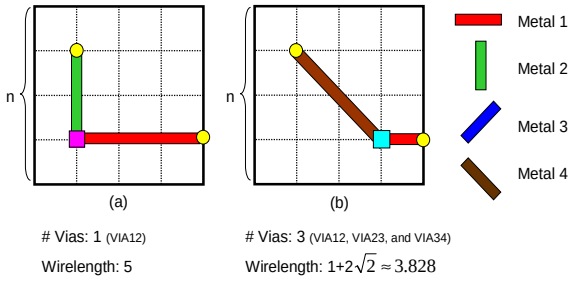

Figure 12: Differences between the Manhattan and the X architecture.

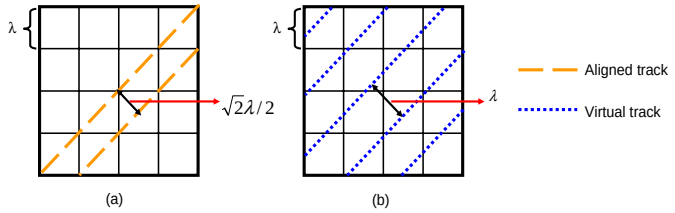

Figure 13: Virtual tracks to meet the minimum spacing rule (λ).

jogs, which are used on the non-preferred direction routing layer and thus include no vias, to connect the end points to the nearest grid.

In this paper, we propose a fast track assignment heuristic for long diagonal segments. After the coarsening stage, we get several long diagonal segments. To simplify the track assignment problem, we track-assign only segments which span more than one complete diamond-shaped global cell and delegate short segments to the detailed router. The track assigner works on a trapezoid-shaped row or column of the diamond-shaped global cell array one at a time (see Figure 14). Each trapezoid-shaped row (column) is called a *trapezoid panel* .

Let T be the set of tracks inside a trapezoid panel. Let ℓ be the set of segments which need to be track assigned in this panel. Each track $t \in T$ can be represented by its set of constituent contiguous intervals. Denoting these intervals by x_i, we have $t \equiv \biguplus x_i$, Each of this x_i is either

- a blocked interval, where no segment from ℓ can be assigned,
- an occupied interval, where segments from ℓ has been assigned, or
- a free interval, where no segment from the set ℓ has yet been assigned.

A segment $seg \in \ell$ is called a left (right) segment, if the left- (right-) end terminal is in the left (right) zone. If a segment is said to be assignable to $t \in T$, $t \equiv \biguplus x_i$, iff $x_i \cap seg \neq \emptyset$, it implies that either x_i is a free interval or it is an interval occupied by a segment of the same net. Thus, a trapezoid-shaped track assignment problem can be defined as follows:

Trapezoid-Shaped Track Assignment Problem: Given a set of tracks T in a trapezoid panel and a set of segments ℓ, and a cost function $F : \ell \times T \longrightarrow N$, which represents the cost of assigning a segment to a track, find an assignment that minimizes the sum of the costs of the assignment.

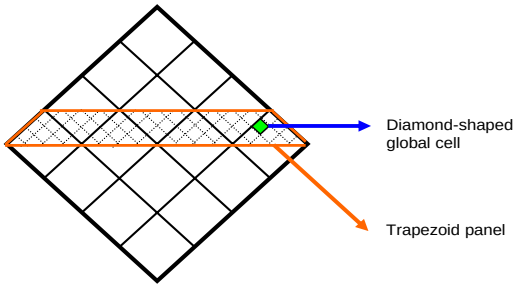

Figure 14: Trapezoid panel.

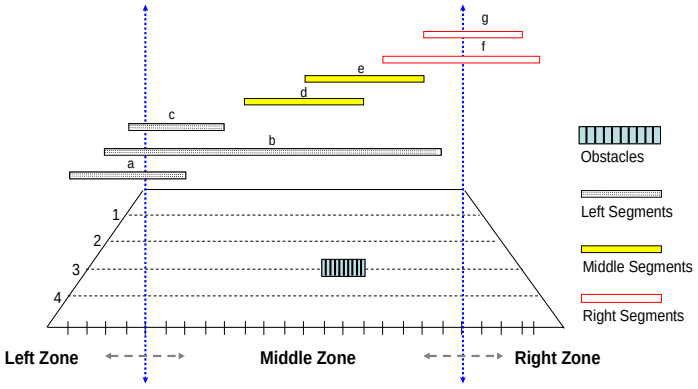

Figure 15: Example of the trapezoid-shaped track assignment problem.

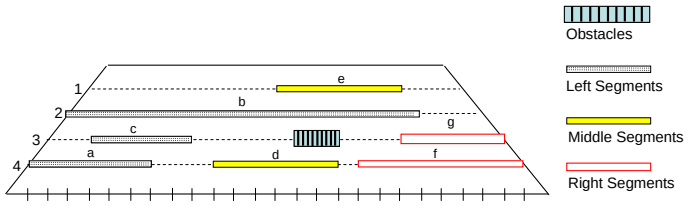

Figure 16: Solution to the trapezoid-shaped track assignment problem given in Figure 15.

In our implementation, we have considered the basic cost metrics such as the planar anchoring cost and the track and via obstruction cost defined in [2]. To better utilize the tracks in the trapezoid panel, we will try to assign the left and right segments to the tracks in the bottom-up fashion. After these segments have been assigned, other segments are assigned by the well-known left-edge algorithm [13] for efficient track assignability.

An example is shown in Figure 15, and the solution is shown in Figure 16. After the track-assignment phase, we use the short wrong-way jogs, which include no vias, to connect the two-end terminals to their nearest grid point. After that, we can perform point-to-path maze routing to complete both end points, which span at most two global cells.

4. EXPERIMENTAL RESULTS

We implemented our multilevel X routing system in the C++ language on a 1 GHz SUN Blade 2000 workstation with 1GB memory. We compared our results with [14] based on the six benchmark circuits provided by the authors (see Table 1 for the benchmark circuits). In Table 1, "Circuits" denotes the names of the circuits, "Size" gives the layout dimensions, "#Layers" denotes the number of routing layers used, and "#Nets" represents the number of two-pin connections after net decomposition.

Experimental results on wirelength, the number of vias, the routing completion rate, and average net delay are listed in Table 2, where "D_{avg}" represents the average net delay. To perform experiments on timing-driven routing, we used the same resistance and capacitance parameters as those used in [14] for comparison. A via is modeled as the π-model circuit, with its resistance and capacitance being twice those of a wire segment. Compared with [14], the experimental results show that our multilevel X router reduced the wirelength and average delay by about 18.7% and 8.8% with similar routability and number of vias in shorter running time. The improvement of via count is not as we expected, because the work of [14] uses four layers for track assignment which reduces lots of vias, and our multilevel X router just use top-two layers instead.

It should be noted that the 18.7% improvement in wirelength is significantly better than the 11% improvement obtained by Toshiba's tool for routing its TC90400XBG digital-media application processor, as mentioned earlier. The difference also implicitly reveals the effectiveness of our multilevel routing.

The experimental results also reveal the effectiveness of the interme-

Circuits	Results of [14]					Our Results				
	Wirelength	#Vias	Cmp. Rates	D_{avg}	Run-Time	Wirelength	#Vias	Cmp. Rates	D_{avg}	Run-Time
S5378	8.4e7	7451	99.8%	1258	10.6	7.2e7	7432	99.8%	1119	10.5
S9234	6.0e7	6239	99.9%	1009	8.1	5.5e7	6323	99.8%	956	7.9
S13207	2.3e8	16003	99.8%	1243	22.6	1.8e8	15897	99.8%	1074	20.9
S15850	2.9e8	19126	99.7%	1253	62.6	2.2e8	18999	99.4%	1178	54.1
S38417	8.0e8	49816	99.8%	1146	71.3	6.1e8	49131	99.0%	1034	59.8
S38584	1.1e9	65798	99.8%	1151	255.6	8.8e8	65018	99.1%	1068	198.1
Comp.	1.19	1.00	1.00	1.09	1.13	1	1	1	1	1

Table 2: Results of wirelength, via counts, completion rate, delay, and run-time comparison.

Circuits	Size (μm)	#Layers	#Nets	#Pins
S5378	4330x2370	4	3124	4734
S9234	4020x2230	4	2774	4185
S13207	6590x3640	4	6995	10562
S15850	7040x3880	4	8321	12566
S38417	11430x6180	4	21035	32210
S38584	12940x6710	4	28177	42589

Table 1: Benchmark circuits.

diate stage of track assignments, because most longer, straight diagonal segments that get track-assigned can take full advantage of the X-architecture. Furthermore, the XST algorithm can also reduce the wirelength and average net delay. Thus, the results show that our new multilevel X routing framework is capable of handling the X-architecture for interconnect optimization.

5. CONCLUSION

In this paper, we have proposed a novel framework for fast multilevel routing for the X-architecture. The experimental results have shown that our approach is very efficient and effective, and that the multilevel framework for the X-architecture is an elegant framework for large SoC, ASIC, and application-specific standard product (ASSP) designs. Our future work lies in an integrated multilevel placement and routing system for the X-architecture.

6. ACKNOWLEDGEMENT

We would like to thank Dr. Cliff Hou, Dr. L. C. Lu, and Mr. Ken Wang of TSMC for very helpful discussions. We also thank anonymous reviewers for their very constructive comments.

7. REFERENCES

[1] http://www.xinitiative.org/

[2] S. H. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, "Track assignment: A desirable intermediate step between global routing and detailed routing," *Proc. of Int. Conf. Computer-Aided Design*, pp. 59–66, 2002.

[3] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd Edition, Springer-Verlag 2000.

[4] Y.-W. Chang, K. Zhu, and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," *Trans. on Design Automation of Electronic Systems*, vol. 5, no. 3, pp. 433–450, 2000.

[5] H. Chen, C. K. Cheng, A. B. Khang, I. I. Mandoiu, Q. Wang, and B. Yao, "The Y-Architecture for on-chip interconnect: Analysis and methodology," *Proc. of Int. Conf. Computer-Aided Design*, pp. 13–19, 2003.

[6] H. Chen, B. Yao, F. Zhou, and C. K. Cheng, "The Y-Architecture: Yet another on-chip interconnect solution," *Proc. of Asia and South Pacific Design Automation Conf.*, pp. 840–846, 2003.

[7] B. Choi, C. Chiang, J. Kawa, and M. Sarrafzadeh, "Routing resources consumption on M-arch and X-arch," *Proc. of Int. Symp. on Circuits and Systems*, 2004.

[8] J. Cong, J. Fang, and Y. Zhang, "Multilevel approach to full-chip gridless routing," *Proc. of Int. Conf. Computer-Aided Design*, pp. 396–403, 2001.

[9] J. Cong, M. Xie, and Y. Zhang, "An enhanced multilevel routing system," *Proc. of Int. Conf. Computer-Aided Design*, pp. 51–58, 2002.

[10] J. Cong and J. Shinnerl, *Multilevel optimization in VLSICAD*, Kluwer Academic Publishers, 2003.

[11] C. S. Coulston, "Constructing exact octagonal steiner minimal trees," *Proc. of Great Lake Symp. on VLSI*, pp. 1–6, 2003.

[12] M. R. Garey, R. L. Graham, and D. S. Johnson, "The complexity of computing steiner minimal trees," *SIAM Journal on Applied Mathematics*, pp. 835–859, 1977.

[13] A. Hashimoto and J. Stevens , "Wire routing by optimizing channel assignment within large apertures," *Proc. of Design Automation Conf.*, pp. 155–169, 1971.

[14] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D. T. Lee, "A fast crosstalk- and performance-driven multilevel routing system," *Proc. of Int. Conf. Computer-Aided Design*, pp. 382–387, 2003.

[15] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with antenna avoidance," *Proc. of Int. Symp. on Physical Design*, pp. 34–40, 2004.

[16] A. B. Kahng, I. Mandoiu, and A. Zelikovsky, "High scalable algorithms for rectilinear and octilinear steiner trees," *Proc. of Asia and South Pacific Design Automation Conf.*, pp. 827–833, 2003.

[17] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling," *IEEE Trans. on Computer-Aided Design*, pp. 777–790, 2002.

[18] C. K. Koh and P. H. Madden, "Manhattan or Non-Manhattan? A study of alternative VLSI routing architectures," *Proc. of Great Lake Symp. on VLSI*, pp. 47–52, 2000.

[19] C. Y. Lee, "An algorithm for path connection and its application," *IRE Trans. Electronic Computer*, EC-10, 1961.

[20] S.-P. Lin and Y.-W. Chang, "A novel framework for multilevel routing considering routability and performance," *Proc. of Int. Conf. Computer-Aided Design*, pp. 44–50, 2002.

[21] M.Paluszewski, P. Winter, and M. Zachariasen, "A new paradigm for general architecture routing," *Proc. of Great Lake Symp. on VLSI*, pp. 202–207, 2004.

[22] M. R. Stan, F. Hamzaoglu, and D. Garrett, "Non-manhattan maze routing," *Proc. of Brazilian Symp. on Integrated Circuit Design*, pp. 260–265, 2004.

[23] S. Teig, "The X Architecture: not your father's diagonal wiring," *Proc. of System Level Interconnect Predicition*, pp. 33–37, 2002.

[24] Q. Zhu, H. Zhou, T. Jing, X. Hong, and Y. Yang, "Efficient octilinear steiner tree construction based on spanning graphs," *Proc. of Asia and South Pacific Design Automation Conf.*, pp. 687–690, 2004.