Timing-Driven Non-Rectangular Obstacles-Avoiding Routing Algorithm for the X-Architecture

Hsin-Hsiung Huang\textsuperscript{1}, Shu-Ping Chang\textsuperscript{2}, Yu-Cheng Lin\textsuperscript{3}, and Tsai-Ming Hsieh\textsuperscript{4}

\textsuperscript{1}Dept. of Electronic Engineering, Lunghwa Univ. of Science and Technology, Taoyun, Taiwan, \textsuperscript{2}Dept. of CAD, Genesys Logic Company, Taipei Country, Taiwan, \textsuperscript{3}Dept. of Information and Electronic Commerce, Kainan University, Taoyuan, Taiwan, \textsuperscript{4}Dept. of Information and Computer Engineering, Chung Yuan Christian University, Chung-Li, Taiwan

Email: pp022@mail.lhu.edu.tw; Patty.Chang@genesyslogic.com.tw; linyu@mail.knu.edu.tw; hsieh@cycu.edu.tw

Abstract: This study formulates a novel timing-driven rectangular and non-rectangular obstacle-avoiding X-based Steiner minimal tree (TNOA-XSMT) problem. The problem is first studied in the literature, and solved by using a timing-driven routing algorithm. The algorithm, which handles both the rectangular and non-rectangular obstacles, minimizes the maximum source-to-terminal delay. A novel concept of incorporating the virtual nodes during the spanning graph construction is presented to minimize the total wire length. An extension is provided to explain how the algorithm handles the obstacles with any geometric shape. Moreover, an effective and efficient rerouting scheme is adopted to further reduce the delay. Rerouting reduces the maximum source-to-terminals delay by 49.1\%, while increasing the additional total wire length by only 2.5\%.

Key-Words: Global routing, nonrectangular obstacle-avoiding, X-architecture, timing-driven, rerouting

1 Introduction

Routing plays a very important role in physical design, since it is a very complex step in nanometer IC. Modern routers present various challenges, including the novel X-based architecture, the obstacles in SOC design and the timing issues.

Several approaches to construct the Steiner minimal tree with minimization of total wire length have been presented under the $\lambda$ -geometry\cite{1-7}. Coulston\cite{3} presented a two-step algorithm that generates all possible full components and merges them into an optimal tree. Kahng et al.\cite{4} presented the wire length-driven heuristic algorithm that adopts the batched-based triple contraction with complexity $O(n \lg^3 n)$, where $n$ is the number of terminals. Chiang et al.\cite{5} presented the octilinear Steiner tree by using the edge-conversion and Steiner-sliding technique. Teig\cite{6} designed a Toshiba microprocessor by using the octagonal technique, reducing the total wire length, via count and die size, by 20\%, 40\% and 11\%, respectively. Some methods cannot handle obstacles and timing issue.

To consider the obstacles is necessary to the modern routers because the SoC technique applies huge IPs and macros\cite{8-10}. Feng et al.\cite{10} presented a three-step method to construct an obstacle-avoiding routing tree that minimizes the total wire length under the $\lambda$-Geometry plane, and can handle non-rectangular obstacles. Wu et al. proposed a graph-based approach, which estimates the wire length with the obstacles, to efficiently construct the routing tree\cite{8}. Some algorithms cannot deal with non-rectangular obstacles, and some algorithms do not consider the timing issues.

Many timing-driven algorithms have been proposed to enhance the performance of the nanometer VLSI design. Many algorithms attempt to reduce detours made during routing\cite{11-16} because these detours result in the large maximum source-to-terminal delays. Huang et al. proposed the timing-driven algorithm, which constructed the obstacle-avoiding routing tree followed by the rerouting for the critical paths with obstacles, to improve the performance of the tree\cite{11}. Pan et al. constructed the initial routing tree and performed the post-processing (branch-moving) to improve the delay of the critical path. Lin et al.\cite{14} proposed a recalling function to update the critical paths and Hu et al.\cite{16} presented the concept of soft edge to move the Steiner points. However, some algorithms cannot manage the routing with obstacles. Some methods cannot reduce the delay by passively adjusting the critical path.

This study makes the following major contributions. (1) The TNOA-XSMT problem is formulated and solved by using a new proposed algorithm. To the best of our knowledge, no existing literature discusses X-based timing-driven trees with non-rectangular obstacles. (2) The effective rerouting method, which splits violated terminals and merges two sub-trees, is proposed to enhance the timing of the violated terminals. Experimental results indicate that if the initial routing tree is not optimal, then the
rerouting not only minimizes the delay but also reduces the wire length. (3) Non-rectangular obstacles are handled by superimposing a bounding rectangle for the non-rectangular obstacles. The concept of virtual nodes is presented to further reduce the wire length. An extension is shown to reveal that the proposed method can handle obstacles with any geometric shapes.

The remainder of this paper is organized as follows. Section 2 describes the problem, including the motivation, the novel concept for the non-rectangular obstacle and problem definition. Section 3 presents the our algorithm. Experimental results and conclusions are shown in section 4 and 5.

![Fig. 1. Routing with non-rectangular obstacles.](image1)

Fig. 1. Routing with non-rectangular obstacles.

![Fig. 2. Terminals inside the virtual obstacles.](image2)

Fig. 2. Terminals inside the virtual obstacles.

## 2 Preliminary

This section describes the motivation and the novel concept for non-rectangular obstacle. Finally, the TNOA-XSMT problems are formulated.

### 2.1 Motivation

Fig. 1 illustrates that the short wire length is obtained by the modern router, which can handle the routing with the non-rectangular obstacles. To the best of our knowledge, no routers for X-architectures have yet been reported in the literature. For the Fig. 1(a), the traditional router, which treats a non-rectangular obstacle as a rectangular obstacle, has the results in Fig. 1(b) but the total wire length is long. However, the short total wirelength in Fig. 1(c) is obtained by the modern router which can handle routing with non-rectangular obstacles. We observe that the algorithm can handle the routing with non-rectangular are necessary.

Besides, we consider the terminals inside the virtual obstacle which is the minimal bounding box superimposing the non-rectangular obstacles (Fig. 2(a)). If we only take the corner points of the virtual obstacles into the spanning graph construction, we have the long wire length (Fig. 2(b)). However, the short wire length is obtained in Fig. 2(c). Two observations motivate us to develop the modern X-architecture router to handle the timing-driven routing with the non-rectangular obstacles.

### 2.2 Frame-based Transformation and A-Shaped Pattern Routing

To simplify, the frame-based transformation which finds the minimum bounding box containing the non-rectangular obstacle (Fig. 3(a)), is proposed to transform the non-rectangular obstacles into the virtual obstacles in Fig. 3(b). Because the X-based routing is used in the paper, only the isosceles right triangle is listed in the experimental results. In fact, the concept is suite for not only the triangles but also the hexagon, octagon, and the polygon. Furthermore, we construct the spanning graph for the rectangles, virtual obstacles and terminals. How about the terminals inside the virtual obstacles?

A novel A-shaped pattern routing, which adds two projected virtual nodes onto the non-rectangular obstacles, is used. First, we construct the spanning graph for the terminals inside the virtual obstacles and the corner points of the virtual obstacles (see Fig. 3(c)). Second, the additional edges, which connect the virtual nodes and the corresponding terminals inside the virtual obstacles, are added into the spanning graph (Fig. 3(d)). Finally, we can construct the better routing tree like Fig. 1(c). Generally, if there are \( v \) terminals inside the obstacles, we need \( 2v \) additional points and \( 2v \) edges in the spanning graph. Hence, the A-shaped patter routing which needs the less memory is efficient.

### 2.3 Problem Definition

The similar concept in [11] is used to obtain a timing-driven routing tree with the maximum source-to-terminal delays (\( D_{\text{max}} \) for short), and we rerouted the critical path which their delays are larger than the delay threshold \( d_{\text{thd}} = D_{\text{max}} \times r \), where \( r \) is the delay ratio. A larger (smaller) ratio denotes looser (tighter) timing constraints. After rerouting which an edge of the routing tree is removed and then two sub-trees are merged, we have the timing-driven routing tree. Hence, the problem is defined as follows,

Given a set of the rectangular obstacles \( \{b_1, b_2, \ldots, b_n\} \) and the non-rectangular obstacles \( \{o_1, o_2, \ldots, o_m\} \), and a set of terminals \( \{t_1, t_2, \ldots, t_k\} \) with the source \( s \) under the user-defined delay threshold \( (d_{\text{thd}}) \), the objective is to construct the X-architecture routing tree which minimizes the \( D_{\text{max}} \) among all obstacles.
3 Timing-Driven Routing Tree

This section discusses two steps and an example.

3.1 Construct an Initial Routing Tree

Two novel ideas are used to handle the terminals inside the non-rectangle obstacles. One is frame-based transformation which superimposes a minimal square bounding rectangle for the non-rectangular obstacle (to generate a virtual obstacle) to effectively handle non-rectangular obstacles. The other is the A-shaped pattern routing which generates two virtual nodes per terminal inside the virtual obstacles to construct the spanning graph for the virtual obstacles. Furthermore, a feasible X-based routing tree is generated by transforming the edges of the minimal spanning tree, which is constructed in the spanning graph.

3.2 Reroute for All Violated Terminals

To improve the performance of the initial routing tree, we reroute the terminals with delays over the user-defined delay threshold. The concept is similar to [11]. First, we remove the selected edge between two nodes, which one is less than the user-defined delay and the other is larger than the user-defined delay. Hence, two sub-trees are obtained after removing the dedicated edge. Second, to improve the delay, we merge two sub-trees by adding the proper edges which reduce the critical delay.

A simple example is used to explain the algorithm. The frame-based transformation and A-shaped pattern routing are used to construct the graph in Fig. 4(a). Then, we obtain feasible X-based routing tree in Fig. 4 (b). For the initial routing tree, the delays of the terminals marked in gray colors in Fig. 4(c) are over 2590.53 fs and a selected edge (marked “×”) is removed to produce two sub-trees. In Fig. 4(d), two sub-trees are merged by the selected edge, which connects the one sub-tree to the node close to the source. Fig. 5 shows the TNOA-XSMT algorithm.

Algorithm: TNOA-XSMT

Input: \( n \) terminals, a source, the \( m \) rectangular and \( w \) non-rectangular obstacles, and delay threshold.

Output: A timing-driven X-based routing tree.

begin
1 Construct the X-based routing tree with obstacles
   1.1 Build the spanning graph;
   1.2 Obtain the minimal spanning tree;
   1.3 Construct the X-based routing tree;
2 Reroute the violated terminals;
   2.1 Split the routing tree by the delay threshold;
   2.2 Merge two sub-trees into one tree;
   2.3 Goto Step 2.1 and 2.2 iteratively until all violated terminals are rerouted once;
end.

Fig. 5. Algorithm for TNOA-XSMT problem.

4 Experimental Results

All experiments were implemented by the C++ language on the Intel Core2 CPU 1.86GHz 1.87GHz machine with the 3GB memory. The objective was to minimize the \( D_{\text{max}} \) by the modified delay model [14] under the 0.18\( \mu \)m technology. Our algorithm which is independent to the delay model can be effective to improve the \( D_{\text{max}} \). Due space limitations, only the result of \( r=0.7 \) is addressed. In Table 1, “Term”, “OB1” and “OB2” denote the numbers of terminals, rectangular obstacles and non-rectangular obstacles, respectively. Because of the lack of source information, Huang’s approach [11] was adopted to test the algorithms under three sources.

First, the delays produced by the two proposed methods were compared. Because there is no algorithm available for TNOA-XSMT, only the two proposed algorithms without and with rerouting were compared. In Table 1, \( T_i \) and \( T_r \) represent the results of the initial tree (step 1 in Fig. 5) and the rerouted tree (step 2 in Fig. 5). “\( L_{\text{tot}} \)” and “\( D_{\text{max}} \)” denote the total wire length and the maximum source-to-terminal delay, respectively. \( T_i \) has a delay 49.1% less than \( T_r \), and 2.5% extra total wire length. Moreover, rerouting reduced the total wire length of T6, T8 and T9. These results indicate that the proposed rerouting technique can minimize both the \( L_{\text{tot}} \) and \( D_{\text{max}} \).
Second, we investigate the rerouting effect on the detour routing, i.e., \( D_{\text{max}} \). Fig. 6 shows the rerouting effect on delay, indicating that the detour in routing (the red line) is reduced. Because of paper limit, Table 1 only shows the result of source type I. In fact, our method stably reduced the \( D_{\text{max}} \) under three source types. If the rerouting does not reduce the \( D_{\text{max}} \) of the initial tree, the initial result is retained.

5 Conclusion

This study formulates a novel TNOA-XSMT problem. Additionally, this work presents an effective timing-driven algorithm that minimizes the \( D_{\text{max}} \) to construct an X-based routing tree with non-rectangular obstacles. Compared with the result without rerouting, the experimental results indicate that the \( D_{\text{max}} \) with rerouting is improved by 49.1% with only 2.5% additional wire length.

### References:


