Timing-Driven Routing for Symmetrical-Array-Based FPGAs *†‡

Kai Zhu[1], *Yao-Wen Chang*[2], *and D. F. Wong*[3]

[1]Triscend Corp., 301 N. Whisman Rd., Mountain View, CA 94043, USA

[2]Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan

[3]Department of Computer Sciences, University of Texas at Austin, Austin, Texas 78712, USA

Abstract

In this paper, we present a timing-driven global router for symmetrical-array-based FPGAs. The routing resources in the symmetrical-array-based FPGAs consist of segments of various lengths. Researchers have shown that the number of segments, instead of wirelength, used by a net is the most critical factor in controlling routing delay in an FPGA. Thus, traditional measure of routing delay based on the geometric distance of a signal is not accurate. To consider wirelength and delay simultaneously, we study a model of timing-driven routing trees, arising from the special properties of FPGA routing architectures. We explore the complexity of the routing-tree problem and present efficient and effective approximation algorithms for the problem. Based on the solutions to the routing-tree problem, we present a global-routing algorithm which is able to utilize various routing segments with global consideration to meet the timing constraints. Experimental results on benchmark circuits show that our approach is promising.

1 Introduction

The symmetrical-array architecture is used in several popular commercially available FPGAs [2, 24]. A symmetrical-array-based FPGA consists of a two-dimensional array of logic modules interconnected by vertical and horizontal routing channels [4]. (See Figure 1 for a typical FPGA model.) The logic modules, denoted by L in Figure 1, can be customized to realize logic functions. The routing channels comprise general routing resources used to connect logic-module pins. An intersection area of a horizontal and a vertical routing channels is referred to as a *switch module*, denoted by S in Figure 1. The switch modules house programmable switches. FPGA routing uses programmable switches (inside S) to make connections. The switches usually have high resistance and capacitance, and thus incur significant delays. To improve circuit performance and maintain reasonable routability simultaneously, the routing channels are usually segmented, and thus routing tracks consist of wires with a versatile set of lengths [24] (see Figure 2). Longer segments are intended for high fan-out, timing-critical signal nets. On the other hand, shorter segments are intended for short connections to avoid wasting routing resources. To achieve the goal of improving circuit performance without much sacrifice on routability, it is essential for a routing algorithm to utilize these various routing segments effectively.

Many routing algorithms for symmetrical-array-based FPGAs have been reported in the literature [1, 5, 6, 10, 16, 17, 18, 21]. Most of these algorithms, however, either aim at routability only, without considering timing constraints [5, 6, 21], or consider only one type of wire segments [1, 5, 6, 16, 18]. Research on timing-driven routing for symmetrical-array-based FPGAs with multi-length segments is very limited. An FPGA routing algorithm considering routability, delay, and multi-length segments is described in [17]. The algorithm is an extension of the detailed-routing algorithm of [5] to handle the selection of different segments for detailed routing. However, a drawback of the approach in [17] is that the choice of segments is restricted to the routing paths defined by a conventional global router, and thus it is hard for the router to utilize the segments efficiently. Furthermore, the timing constraints, which are usually specified as arrival times on primary inputs and required times on the primary outputs [13], are not completely considered in the routing algorithm.

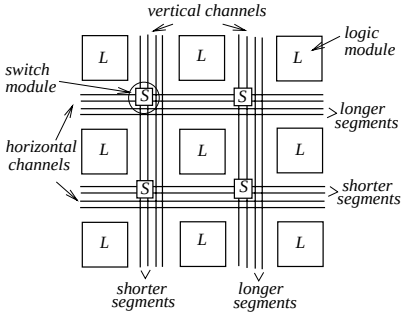

Figure 1: The symmetrical-array-based FPGA model.

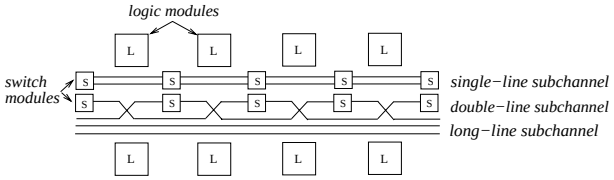

Figure 2: Detailed structure of a routing channel.

Researchers have shown that, due to the segmented architectures of FPGA routing channels, the number of segments, instead of wirelength, used by a net is the most critical factor in controlling routing delay in an FPGA [9, 14]. Thus, traditional measure of routing delay based on the geometric distance of a signal is not accurate for the FPGA with multiple segment lengths. As shown in Figure 3, both nets 1 and 2 have the same geometric distance. However, the signal delay of net 2 is significantly larger than that of net 1 because net 2 uses more switches to make the connection.

To consider wirelength and delay simultaneously, we study a model of timing-driven routing-tree problem, arising from the special properties of FPGA routing architectures. We explore the complexity of this problem and present efficient and effective approximation algorithms for the problem. Based on the solution to the routing-tree problem, we present a timing-driven global-routing al-

*The portion of the work done by Kai Zhu was performed when he was with The University of Texas at Austin.

†The work of Yao-Wen Chang was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC-87-2215-E-009-041.

‡The work of D.F. Wong was partially supported by the Texas Advanced Research Program under Grant No. 003658288.

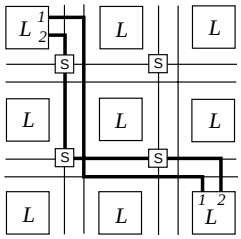

Figure 3: Routing on different types of segments. Net 2 has larger delay than Net 1 because it uses more switches.

gorithm for symmetrical-array-based FPGAs. The global-routing algorithm has the following distinct features, compared with previous work:

- The utilization of routing resources is considered with a global viewpoint. The routing algorithm uses a hierarchical top-down approach and considers all available routing resources in assigning routing paths to nets.
- The algorithm explicitly includes timing constraints as its inputs and aims at routing all the nets to meet the constraints. To perform timing-driven routing, the timing constraints are transformed into the delay bounds on the source-to-sink paths of nets. Further, the routing algorithm can be used as a direct follow-up to timing-driven placement [12, 20].
- Detailed routing can be performed simultaneously with global routing. All nets are routed at the same time, and thus the net ordering problem can be avoided during both global- and detailed-routing stages.

We have performed experiments on benchmark circuits. The results show that our timing-driven router is very promising.

2 Problem Formulation

A horizontal (vertical) channel is the routing space between two adjacent rows (columns) of logic modules. A routing channel is divided into a set of *subchannels*. Subchannels are distinguished by the relative length of their segments, and each subchannel consists of a set of equal-length routing segments. Figure 2 shows an example of a horizontal channel with three subchannels, namely *single-line subchannel*, *double-line subchannel*, and *long-line subchannel*. The single-line subchannel consists of single-length segments which form a grid of horizontal and vertical lines that intersect at switch modules. The double-line subchannel contains double-length segments composed of a grid of segments twice as long as the single-length ones. The long-line subchannel contains segments that run the entire vertical or horizontal channel. This model of channel segmentation is used in popular commercial FPGAs [2, 24].

The timing constraints on a circuit are specified as the *arrival times* at the primary inputs or outputs of storage elements, and the *required times* at the primary outputs or the inputs to storage elements [13]. In timing-driven physical layout, the timing constraints can be transformed into the delay bounds on nets [10, 12]. Every net consists of a *source* pin and a set of *sink* pins. The timing constraints on a net are specified as the delay bound from the source to each sink. The goal of timing-driven routing is thus to route the nets so that the delay constraints are satisfied.

To perform timing-driven routing, the net delay bounds need to be transformed into physical layout constraints. It is shown in [14] that the number of segments (and thus the number of switches) a net has to pass through is the most critical factor in controlling routing delay in an FPGA. The delay model based on the number of switches used by a net is sufficiently accurate for the purpose of timing-driven routing. See Example 1 for an example of delay modeling.

Example 1 *In this example, we demonstrate, based on the Elmore delay model [8], how a delay constraint on a routing path can be transformed into the upper bound on the number of switches used.*

Consider a chain of k switches with driver resistance R_d at the source and load capacitance C_L at the sink. Each switch has on-resistance R_0 and capacitance C_0, and is modeled by a two-terminal L-model equivalent circuit as shown in Figure 4(a). The equivalent circuit for the chain of k switches is shown in Figure 4(b). Based on the Elmore delay model, the source-to-sink delay of the chain is given by:

$$\begin{aligned} d(s,t) &= \sum_{u \in \delta(s,t)} R_u C_u \\ &= (R_d + R_0)(kC_0 + C_L) + R_0((k-1)C_0 + C_L) + \\ &\quad \ldots + R_0(C_0 + C_L) \\ &= \alpha + \beta k + \gamma k^2, \end{aligned} \tag{1}$$

where $\alpha = R_d C_L, \beta = R_d C_0 + R_0 C_L + \frac{1}{2} R_0 C_0$, and $\gamma = \frac{1}{2} R_0 C_0$. Here, $\delta(s,t)$ is the set of nodes along the path from source s to sink t, excluding s.

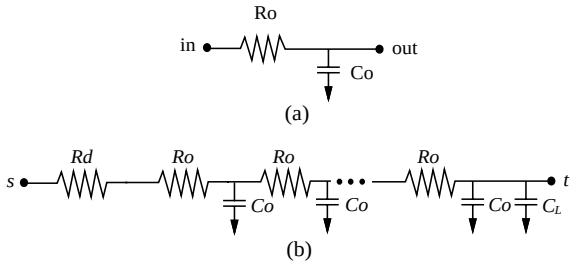

Figure 4: (a) Equivalent circuit for a switch. (b) Equivalent circuit for a chain of switches.

Given a two-terminal net N, the source-to-sink delay bound B of N can be converted into an upper bound K on the total number of switches used in routing N by modeling N as a chain of switches. From Equation (1), we can solve the inequality $\alpha + \beta k + \gamma k^2 \leq B$ such that the left-hand side of the inequality is maximized. This gives

$$K = \left\lfloor \frac{-\beta + \sqrt{\beta^2 - 4\gamma(\alpha - B)}}{2\gamma} \right\rfloor.$$

It is not difficult to generalize the above modeling to multi-terminal nets, and we shall omit the discussion.

Based on such a delay model, the timing constraints of a net can therefore be transformed into the upper bounds on the number of switches used by source-to-sink paths of the net. Note that the number of switches used along the routing paths of the net is known if the global-routing paths of a net is specified in terms of subchannels.

We formulate the timing-driven global-routing problem for the symmetrical-array-based FPGAs as follows:

- **The FPGA Timing-Driven Global-Routing Problem (FPGA-TGRP)**
  Instance: Given an FPGA architecture, a netlist of the placed circuit, and timing constraints specified as the upper bounds on the number of switches used along all source-sink pairs of nets.
  Objective: Determine the global-routing paths (a sequence of subchannels) for each net so that the net delay bounds are satisfied.

3 The Routing Algorithm

3.1 Overview

The routing algorithm is based on the hierarchical top-down strategy for traditional ASICs [15, 19], with several distinguished features to deal with the special properties of FPGA routing architectures.

We shall first review the generic hierarchical approach and then present the unique features in our router. The generic hierarchical approach proceeds as follows. Starting with the entire circuit, a cut line is chosen in a certain way (such as min-cut [3]) to divide the circuit into two parts. Across the cut line there are *routing sections* which represent routing spaces on the chip. (See Figure 5 for an illustration of the terminology.) A routing section is usually a channel with a routing capacity specified by the channel width. Each connection crossing the routing section is assigned a cost defined by total wire length and channel density. The global routing at each hierarchical level assigns connections to routing sections based on a particular algorithm such as linear assignment [15, 19]. After finishing global routing at this hierarchical level, both subcircuits separated by the cut line are routed by the same method recursively. The algorithm terminates when the subcircuits are small enough so that they can be solved easily.

To deal with the timing-driven global routing for symmetrical-array-based FPGAs, we add the following distinguished features to the above-mentioned generic hierarchical approach.

1. Cut lines are either horizontal or vertical. A routing section on a cut line is a subchannel, instead of a channel. Therefore, the global routing will automatically choose appropriate subchannels for nets at each hierarchical level.
2. A routing tree is constructed for each net. Here, a routing tree of a net is a spanning tree connecting all pins of the net, subject to the constraints that all paths from the source to sinks must satisfy their delay bounds. We formulate in Section 3.2 a timing-driven routing-tree problem to consider the constraints.
3. After constructing a routing tree, the net delay bounds need to be distributed among the edges of the routing tree in order to perform hierarchical routing. The algorithm for distributing delay bounds is described in Section 3.3.
4. A new cost function for the linear assignment should be defined to consider timing constraints. We define the cost function in Section 3.4.
5. At each hierarchical level, the delay bounds for connections crossing a cut line need to be redistributed among subcircuits for the next level of hierarchical routing. The redistribution of delay bounds should facilitate the routing at the subsequent hierarchical routing levels. Delay-bound redistribution is discussed in Section 3.5.

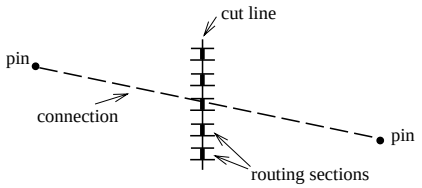

Figure 5: Terminology used in the hierarchical routing.

3.2 Timing-Driven Routing Trees

The goal of timing-driven routing is to construct a routing tree for each net such that all timing constraints are satisfied and the routing cost is minimized (to save routing resources and reduce capacitance). Routing cost usually includes wire lengths and routing congestion. As in conventional technology, geometric distance is a suitable measurement for the routing cost. Due to the segmented structure of routing resources in FPGAs, however, the routing delay is generally not proportional to the geometric distance. Thus, the traditional measure of routing delay based on the geometric distance of a signal is not accurate for an FPGA with multiple segment lengths. To perform timing-driven routing, we formulate the following routing-tree problem with two independent weights, one for the routing cost, and one for the routing delay. It will be clear later that solutions to the problem pave the way to the development of our timing-driven router.

- *The Bounded-Delay Minimum-Cost Spanning Tree Problem* (*BDMCSTP*)
  Instance: Given a graph $G = (V, E)$, $V = \{s, t_1, \ldots, t_{|V|-1}\}$, a routing cost function $r(e) \in \mathbf{Z}^+$ and a delay cost function $d(e) \in \mathbf{Z}^+$ for each $e \in E$, and a delay bound $B(t_i) \in \mathbf{Z}^+$ for each simple path from s to t_i.
  Objective: Find a spanning tree T for G satisfying $\sum_{e \in p(s,t_i)} d(e) \leq B(t_i)$ for every simple path p from s to t_i in T such that $\sum_{e \in T} r(e)$ is minimized.

Let the decision version of the BDMCSTP be *the Bounded-Delay Bounded-Cost Spanning Tree Problem* (*BDBCSTP*).

3.2.1 Complexity of the Routing-Tree Problem

In this subsection, we shall study the complexity of the above problem. The BDBCSTP and BDMCSTP can be trivially solved in polynomial time in the following two special cases: If $\min \sum_{e \in p(s,t_i)} d(e) = B(t_i)$ for every simple path p from s to t_i, the solution is a shortest-paths tree [7] of G, rooted at s and measured on the delay cost d; if $B(t_i)$ is sufficiently large for each i, the solution is the minimum spanning tree of G, measured on the routing cost r. However, we can show that the satisfiability problem of boolean formulas in the 3-conjunctive normal form, 3SAT, is polynominally reduciable to the BDBCSTP, i.e., 3SAT $\leq_p$ BDBCSTP. Since 3SAT is NP-complete [11], the BDBCSTP in general is NP-hard. More specifically, we have the following theorem.

Theorem 1 *The BDBCSTP is NP-complete.*

Corollary 1.1 *The BDMCSTP is NP-hard.*

Since the general BDMCSTP is NP-hard, whether there exists a polynomial-time algorithm for the problem is an open problem. We are thus interested in finding an approximation algorithm with a guaranteed performance bound. However, we can show that if there exists a polynomial-time algorithm $\mathcal{A}$ for the BDMCSTP without triangle inequality, then $\mathcal{A}$ can be used to solve the 3SAT problem, which is a contradiction if $P \neq NP$. Specifically, the following theorem holds.

Theorem 2 *If $P \neq NP$ and $\rho \geq 1$, there is no polynomial-time approximation algorithm with performance bound ρ for the general BDMCSTP.*

Usually, the triangle inequality for the routing cost holds in practical applications. Given a graph $G = (V, E)$, let the routing costs of its minimum spanning tree measured on r, MST_r, and an optimal tree, T_{BDMCST}, to the BDMCSTP on G be Φ_{MST_r} and Φ_{BDMCST}, respectively. We have the following theorem:

Theorem 3 $\Phi_{MST_r} \leq \Phi_{BDMCST} \leq (|V| - 1)\ \Phi_{MST_r} \leq (|V| - 1)\ \Phi_{BDMCST}$, *if the routing cost satisfies the triangle inequality.*

3.2.2 The Timing-Driven Routing-Tree Algorithm

Since the general BDMCSTP is NP-hard, we resort to a heuristic algorithm to obtain efficient solutions.

For timing-driven global routing, we use the following cost functions. The routing cost of an edge $e = (u, v)$ is defined as the Manhattan distance

$$r(e) = |x_u - x_v| + |y_u - y_v|, \tag{2}$$

where (x_u, y_u) and (x_v, y_v) are the coordinates of the pins u and v, respectively. The delay measurement is given by

$$d(e) = \max\{d_{min}, \alpha r(e)\}, \tag{3}$$

where d_{min} is the minimum number of switches required to connect u and v (we will discuss d_{min} in details later), and $0 < \alpha \leq 1$ is a constant which satisfies the following inequality

$$D_{G,d}(s, t_i) \leq B(t_i), \quad \forall t_i \in V, \tag{4}$$

and at the same time maximize the left-hand side of the above inequality; here $D_{G,d}(s, t_i)$ is the delay cost of the shortest path from

s to t_i on G. Recall that the minimum length of routing segment is the span of one logic module, thus the routing cost function $r(e)$ in the delay function $d(e)$ in Equation (3) represents the maximum possible delay between two pins u and v. The constant α is used to capture the usage of longer routing segments for satisfying stringent delay bounds. Smaller α gives smaller delay cost $d(e)$, and it implies that the routing between u and v needs to use longer segments. The value of α is computed by Inequality (4) on all source-sink pairs of the net. Thus, for the source-sink pairs for which Inequality (4) is strictly less than the corresponding delay bound $B(t_i)$, there will be more alternatives for the routing to meet the delay bounds.

We now discuss delay lower bound d_{min}. To compute d_{min}, we construct a graph H as follows. Represent each pin and routing segment by a vertex. Connect two vertices by an edge with unit weight if there is a switch between the two vertices. That is, the switch either connects a pin with a routing segment, or connects a routing segment to another segment through a switch module. Such a graph H is referred to as a *connectivity graph*. The delay bound d_{min} is the length of the shortest path between u and v on H, and can be computed by a shortest-path algorithm [7]. Notice that since any pin needs to connect to a routing track, it requires at least two switches to connect two pins. Thus $d_{min} \geq 2$. The connectivity graph will be used again in defining the cost function for linear assignment in Section 3.4.

With the cost functions (2) and (3), we use the following algorithm to construct bounded-delay routing trees. The algorithm, $PrimBDRT$, follows the approach of the Prim's minimum-spanning-tree construction [22]. We grow a tree $T = (V_T, E_T)$ incrementally, starting from the source s. At each step, we choose an edge $e = (u, v)$, where $u \in V_T$ and $v \in V \backslash V_T$, such that the routing cost of the edge is minimum and the delay constraint from s to v is satisfied. If no such an edge exists, we find the first vertex u' along the path in T from u to s such that $D_{T,d}(s, u') + D_{G,d}(u', v) \leq B(v)$, where $D_{G,d}(x, y)$ is the delay cost of the shortest path from x to y on G, and then add the nodes and edges on the shortest path (measured on delay cost d) from u' to v into T. Since adding the path could violate the tree property, we thus compute the shortest-paths tree of T, measured on delay. Figure 6 summaries the algorithm. The time complexity of $PrimBDRT$ is $O(|V|^3)$. Based on Theorem 3, we can show that the algorithm enjoys the same performance bound if the triangle inequality holds for the routing cost. (Note that the routing cost defined in Equation (2) satisfies the triangle inequality.)

Corollary 3.1 *$\Phi_{PrimBDRT} \leq (|V| - 1)\Phi_{BDMCST}$ if the routing cost satisfies the triangle inequality.*

3.3 Delay-Bound Distribution

In this subsection, we discuss the distribution of delay bounds on source-sink pairs to the edges of a routing tree.

We first introduce a few terms. Let the edge between the sink t_i and its parent be e_{t_i}. For any sink t_i, there is a unique path from the source s to t_i in T. Let $b(e_{t_i})$ be the delay bound allocated to the edge e_{t_i}. The *slack* of a sink t_i, denoted by S_{t_i}, is defined as the difference between the delay bound $B(t_i)$ and the sum of delay bounds of all edges along the path from s to t_i, i.e., $S_{t_i} = B(t_i) - \sum_{e \in p(s,t_i)} b(e)$, where $p(s, t_i)$ is the path from s to t_i in T. The slack for the source s is defined as zero. Figure 7(a) shows a routing tree with delay bounds on the sinks, a distribution of delay bounds on the edges, and the corresponding slacks.

The delay-bound distribution is to allocate the delay bounds on the edges of the routing tree T such that for every sink t_i in T, the slacks $S_{t_i} \geq 0$. The algorithm for delay-bound distribution is listed in Figure 8. First, the algorithm allocates to every edge e_{t_i} in T an initial delay bound $b(e_{t_i}) = d(e_{t_i})$, i.e., the associated edge delay cost (line 1). Note that with such initial delay bound allocation, the slacks of all sinks in the routing tree T constructed by the PrimBDRT algorithm in Section 3.2 are guaranteed to be non-negative. After the initial allocation, it is possible that the delay bounds on the edges can be further relaxed under certain conditions. Let T_{t_i} be the subtree of T rooted at sink t_i. A sink t_i is a *relaxable sink* if the slack of every sink in T_{t_i} is greater than zero. The following theorem gives the necessary and sufficient condition for further relaxing edge delay bounds.

Algorithm: $PrimBDRT(G, B, s)$
Input: $G = (V, E)$; $B(t_i), \forall t_i \in V$; s.
Output: $T = (V_T, E_T)$ `/* bounded-delay spanning tree. */`
`/*` $STP_{G,d}(x, y)$: `the shortest path from` x `to` y `on` G, `measured on delay cost` d `*/`
begin
1 $V_T \leftarrow \{s\}$; $E_T \leftarrow \emptyset$;
2 **while** $(|V_T| < |V|)$ **do**
3 Among all edges $e_i = (u, v) \in E$, $u \in V_T$, and $v \in V \backslash V_T$, satisfying $D_{T,d}(s, u) + d(e_i) \leq B(v)$, pick e such that $r(e) = \min_i r(e_i)$;
4 **if** such an edge exists
5 $V_T \leftarrow V_T \cup \{v\}$; $E_T \leftarrow E_T \cup \{(u, v)\}$;
6 **else**
7 Find the first vertex u' along the path in T from u to s such that $D_{T,d}(s, u') + D_{G,d}(u', v) \leq B(v)$
8 $V_T \leftarrow V_T \cup \{v |$ vertex v in $STP_{G,d}(u', v)\}$;
9 $E_T \leftarrow E_T \cup \{e |$ edge e in $STP_{G,d}(u', v)\}$;
10 $T \leftarrow$ shortest-paths tree of T measured on delay;
11 **Output** $T = (V_T, E_T)$
end

Figure 6: Algorithm for constructing bounded-delay routing trees.

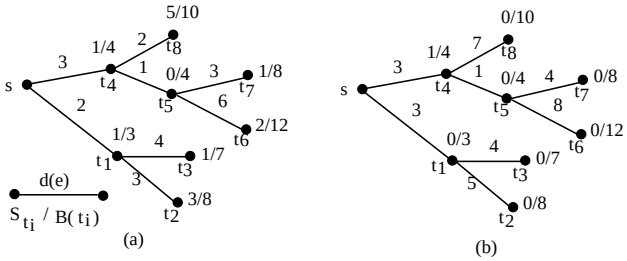

Figure 7: (a) Slack computation. (b) Delay-bound distribution.

Theorem 4 *The delay bound on an edge e_{t_i} can be further relaxed if and only if the sink t_i is a relaxable sink.*

All the relaxable sinks in T can be easily identified in $O(|V|)$ time. After allocating the initial delay bounds on edges, the algorithm picks a relaxable sink t_i, if any, with the minimum slack, and relaxes the the delay bound on e_{t_i} by the amount of S_{t_i} (lines 3-5). The process repeats until there is no relaxable sink in T. Figure 7(b) shows the routing tree after applying the algorithm on that in Figure 7(a). Notice that the slack of t_4 does not change, since t_4 is not a relaxable sink.

Each of Steps 3 and 4 can be done in one traversal of the routing tree T. In each iteration, the number of relaxable sinks is reduced at least by one. Thus there are at most $|V| - 1$ iterations in the while loop, and thus the running time of the algorithm is $O(|V|^2)$.

3.4 Cost Function for Linear Assignment

A cut line at a routing hierarchical level cuts through a set of subchannels. A routing section on the cut line is a subchannel. Let C_{ij} be the cost of routing connection i through subchannel j. The cost C_{ij} consists of three terms:

$$C_{ij} = C_{ij}^{(1)} + C_{ij}^{(2)} + C_{ij}^{(3)}. \quad (5)$$

Algorithm: Delay_Bound_Distribution(T, B, d)
Input: $T = (V, E)$; $B(t_i), \forall t_i \in V$; $d(e), e \in E$.
Output: T with delay-bound distribution.
begin
1 $b(e_{t_i}) \leftarrow d(e_{t_i}), \forall t_i \in V$;
2 **while** (there exist relaxable sinks) **do**
3 Compute slacks $S_{t_i}, \forall t_i \in V$;
4 Label every sink $t_i \in V$ as either relaxable or unrelaxable;
5 Pick a relaxable sink t_i with minimum S_{t_i}, let $b(e_{t_i}) \leftarrow b(e_{t_i}) + S_{t_i}$
end

Figure 8: Algorithm for delay bound distribution.

The first term $C_{ij}^{(1)}$ depends on whether the connection i can reach the subchannel j:

$$C_{ij}^{(1)} = \begin{cases} 0 & \text{if connection } i \text{ can reach subchannel } j, \\ \infty & \text{otherwise.} \end{cases}$$

The reachability can be determined by a breadth-first search on the connectivity graph as defined in Section 3.2.

The second term intends to utilize the routing segments evenly according to the connection length and its delay bound:

$$C_{ij}^{(2)} = a \left| \frac{l_i}{U_i} - L_j \right|,$$

where l_i is the Manhattan distance of the connection i, U_i is the delay bound of connection i, L_j is the length of routing segments in the subchannel j, and $a > 0$ is a constant. Note that $\frac{l_i}{U_i}$ is the average Manhattan distance in routing the connection using at most one switch. Thus the closer $\frac{l_i}{U_i}$ to the length of segments in a subchannel, the lower the cost.

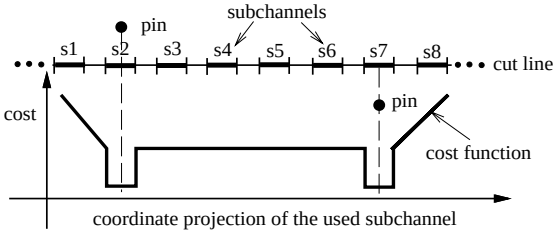

Figure 9: Cost function $C_{ij}^{(3)}$ for linear assignment for a two-pin net. The vertical and horizontal axes denote the cost and the coordinate projection of the used subchannel on the cut line, respectively. The thick curve represents the cost function designed to minimize the number of bends.

A bend in routing a net requires at least one switch. Thus it is preferable for routing a net with fewer bends in order to meet the delay bounds. The third term $C_{ij}^{(3)}$ of the cost function is defined based on this consideration and is illustrated in Figure 9. There are two subchannels (s_2 and s_7) on the cut line through which the connection will have the minimum possible number of bends. Thus $C_{ij}^{(3)}$ has the minimum value at these two subchannels. The cost of routing through the subchannels within the bounding box of the connection $(s_3, s_4, \ldots, s_6)$ is a constant greater than the minimum cost. The costs for routing through subchannels outside the bounding box of the connection (e.g., s_1 and s_8) are proportional to the Manhattan distances between the subchannels and the bounding box.

3.5 Delay-Bound Redistribution

After assigning connections to the subchannels on the cut line, the delay upper bounds for the connections need to be redistributed among the new connections created by the cut line.

We first consider a simple case where the connection crossing the cut line does not share pins with any other connection crossing the cut line. (See Figure 10(a).) Assume the connection i is assigned to the subchannel j. Let $i1$ and $i2$ be the new connections created on the two sides of the cut line. Let l_{i1} (l_{i2}) be the Manhattan distance between pin 1 (pin 2) and the subchannel j. The delay bound U_i of the connection i is redistributed among the new connections $i1$ and $i2$ proportional to their distances to the assigned subchannel, i.e.,

$$U_{i1} = \frac{l_{i1}}{l_{i1} + l_{i2}} U_i,$$

and

$$U_{i2} = \frac{l_{i2}}{l_{i1} + l_{i2}} U_i,$$

where U_{i1} and U_{i2} are the delay bounds for the new connections $i1$ and $i2$, respectively.

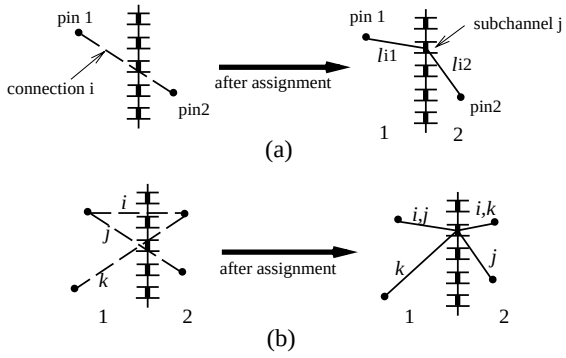

Figure 10: Delay-bound redistribution. (a) Simple case. (b) General case.

In general, connections of a same net crossing the cut line may share pins. If connections which share pins are not assigned to the same subchannel, delay bounds are redistributed for every connection independently as discussed in the above simple case. If several connections which share pins are assigned to a same subchannel, the delay bound redistribution will be different. See Figure 10(b) for an illustration of the redistribution procedure. In Figure 10(b), three connections i, j, and k belong to a same net. Connections i and j share a pin on side 1, and connections i and k share another pin on side 2. All three connections are assigned to a same subchannel. Thus the new connections $i1$ and $j1$ are the same, as well as the new connections $i2$ and $k2$. The delay bound for a new connections created on the side where the connections share a pin is chosen to be the minimum of the delay bounds. In Figure 10(b), the delay bounds for new connections $i1$ ($j1$) and $i2$ ($k2$) are

$$U_{i1} = U_{j1} = min\{\frac{l_{i1}}{l_{i1} + l_{i2}} U_i, \frac{l_{j1}}{l_{j1} + l_{j2}} U_j\}$$

and

$$U_{i2} = U_{k2} = min\{\frac{l_{i2}}{l_{i1} + l_{i2}} U_i, \frac{l_{k2}}{l_{k1} + l_{k2}} U_k\},$$

where U_{i1}, U_{i2}, U_{j1}, and U_{k2} are the delay bounds for the new connections $i1$, $i2$, $j1$, and $k2$, respectively. For a new connection on the side which does not share pins with other connections, the delay bound is the difference between the delay bound for the original connection and that allocated to the new connection on the other side of the cut line. In Figure 10(b), the delay bounds for the new connections $j2$ and $k1$ are $U_{j2} = U_j - U_{j1}$ and $U_{k1} = U_k - U_{k2}$, respectively.

Circuit	# Connections	Circuit Sizes
BUSC	392	12 x 11
X1	436	13 x 12
TooLarge	519	14 x 13
VDA	722	16 x 15
DMA	771	17 x 15
ALU4	851	18 x 16
K2	1256	21 x 19

Table 1: Benchmark circuits.

3.6 Remarks on Detailed Routing

Detailed routing can be easily incorporated into global routing at every hierarchical level. After global routing, the connections assigned to every subchannel are known. For each subchannel, construct a bipartite graph $G = (V_1 \cup V_2, E)$. A connection or a set of connections belonging to a same net and assigned to the subchannel is represented by a node $u \in V_1$. A routing segment in the subchannel is represented by a node $v \in V_2$. There is an edge between a node $u \in V_1$ and a node $v \in V_2$ if and only if all the pins of the connection(s) represented by u can reach the routing segment represented by v. Detailed routing is then solved optimally in polynomial time using a bipartite matching algorithm [7]. If detailed routing cannot route all the connections assigned to a subchannel, the capacity of the subchannel is reduced appropriately and the global routing at this hierarchical level is routed again. The iterative procedure continues until either all connections are routed or a predefined routing completion rate is achieved.

One of the advantages of performing detailed routing and global routing simultaneously is that any overestimation in global routing can be corrected immediately and accordingly.

4 Experimental Results

The proposed timing-driven global-routing algorithm was implemented in C on a SUN SPARC workstation. Table 1 gives the names of the circuits, the numbers of connections (the numbers of source-sink pairs) in each circuit, and the sizes of the circuits (the numbers of logic modules in the FPGA). Note that this set of circuits was used in [9].

The routing resources were chosen as follows. There were three subchannels in each horizontal and vertical channel. The first subchannel contained 8 tracks of single lines. The second subchannel contained 4 tracks of long lines. The third subchannel contained 8 tracks of lines with lengths of 4 units. Thus there were 20 tracks per channel. These parameters of routing segmentation are close to the commercial architecture in [2]. All three types of subchannels can be connected with each other via switch modules.

We randomly picked a source from the pins of a net and made the others sinks. Let $l(s, t_i)$ be the Manhattan distance between the source s and a sink t_i. The delay bound $B(t_i)$ was chosen randomly from the interval $[0.25l(s, t_i), 0.75l(s, t_i)]$. If the delay bound $B(t_i)$ was less than d_{min}, then set $B(t_i) = d_{min}$. Each circuit was routed by the algorithm and the percentage of source-sink pairs which met the delay bounds was computed. The results are shown in the column "Timing-Driven" of Table 2. For the purpose of comparison, we also routed the circuits by the same routing algorithm, with the cost function for the linear assignment to minimize the wire length and the delay bound distributions/redistributions being turned off, i.e., $C_{ij}^{(3)}$ in Equation (5) was set to zero; this leads to a non-timing-driven routing approach [15, 20]. The results are given in the column "Non-Timing-Driven" of Table 2. For all the circuits, the timing-driven routing algorithm achieved substantially better performance in satisfying the delay bounds. The percentages of improvements are listed in the column "Improvement" of Table 2. The results show that an average of 17% improvement is obtained.

Note that we did not compare our results with the previous works in [1, 5, 16, 18], because those previous works consider only one type of wire segments while ours is based on multi-length segments (which is the case in most commercial architectures).

Circuits	Timing-Driven (%)	Non-Timing Driven(%)	Improvement (%)
BUSC	80.6	61.7	30.6
X1	81.2	70.9	14.5
TooLarge	78.4	68.0	15.3
VDA	83.1	71.6	16.1
DMA	78.9	64.6	22.1
ALU4	86.6	74.4	16.4
K2	87.7	80.6	8.8
Average	82.4	70.3	17.2

Table 2: Experimental results for the percentage of nets satisfying the timing constraints.

References

[1] M. Alexander and G. Robins, "New performance-driven FPGA routing algorithms," *Proc. of Design Automation Conf.*, pp. 562–567, June 1995.

[2] AT&T Microelectronics, *AT&T Field-Programmable Gate Arrays Data Book*, Apr. 1995.

[3] M. A. Breuer, "A class of min-cut placement algorithms," in *Proc. of Design Automation Conf.*, pp. 284–290, 1977.

[4] S. Brown, R. Francis, J. Rose, and Z. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Pub., 1992.

[5] S. Brown, J.Ross, and Z. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays," *IEEE Trans. on Computer-Aided Design*, pp. 620-627, May 1992.

[6] Y.-W. Chang, S. Thakur, K. Zhu, and D.F. Wong, "A new global routing algorithm for FPGAs," in *Proc. of Int. Conf. on Computer-Aided Design*, pp. 380–385, Nov. 1994.

[7] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[8] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide band amplifiers," *J. Appl. Phys.*, vol. 19, pp. 55–63, 1948.

[9] B. Fallah and J. Rose, "Timing-Driven Routing Segment Assignment in FPGAs," Canadian Conference on VLSI, 1992.

[10] J. Frankle, "Iterative and Adaptive Slack Allocation for Performance-Driven Layout and FPGA Routing," *Proc. of Design Automation Conf.*, 1992, pp. 536-542.

[11] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, CA, 1979.

[12] P.S. Hauge, R. Nair, and E.J. Yoffa, "Circuit Placement for Predictable Performance," *Proc. of Int. Conf. on Computer-Aided Design*, 1987, pp. 88-91.

[13] R.B. Hitchcock, Sr., G.L. Smith, and D.D. Cheng, "Timing Analysis of Computer Hardware," *IBM J. Research and Development*, vol. 26, No. 1, Jan. 1982, pp. 100-105.

[14] M. Khellah, S. Brown, and Z. Vranesic, "Modelling Routing Delays in SRAM-based FPGAs," *Canadian Conference on VLSI*, 1993.

[15] U. Lauther, "Top-down Hierarchical Global Routing for Channelless Gate Arrays Based on Linear Assignment," *Proc. of IFIP VLSI 87*, 1987, pp.109-120.

[16] Y. -S. Lee and C. -H. Wu, "A performance and routability driven router for FPGA's considering path delay," in *Proc. of Design Automation Conf.*, pp. 557–561, June 1995.

[17] G. Lemieux and S. Brown, "A Detailed Router for Allocating Wire Segments in FPGAs," *ACM Physical Design Workshop*, April 1993.

[18] G. G. Lemienx, S. D. Brown and D. Vranesic, "On two-step routing for FPGAs," *Proc. of ACM Int. Symp. on Physical Design*, pp. 60–66, April 1997.

[19] M. Marek-Sadowska, "Route Planner for Custom Chip Design," *Proc. of Int. Conf. on Computer-Aided Design*, 1986, pp. 246-249.

[20] M. Marek-Sadowska, "Issues in Timing Driven Layout," in *Algorithmic Aspects of VLSI Layout, Lecture Notes Series on Computing*, D.T Lee and M. Sarrafzadeh ed., World Scientific Pub., 1993.

[21] M.Palczewiski, "Plane Parallel A* Maze Router and Its Application to FPGAs," *Proc. of Design Automation Conference*, 1992, pp. 691-697.

[22] A. Prim, "Shortest connecting networks and some generalizations," *Bell System Tech. J.*, pp. 1389–1401, 1957.

[23] S. Thakur, Y.-W. Chang, D. F. Wong, and S. Muthukrishnan, "Algorithms for an FPGA switch module routing problem with application to global routing," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 1, pp. 32–47, Jan. 1997.

[24] Xilinx Inc., *The Programmable Logic Data Book*, 1996.