Graph Matching-Based Algorithms for Array-Based FPGA Segmentation Design and Routing*

Jai-Ming Lin[1], *Song-Ra Pan*[2], *and Yao-Wen Chang*[3]

[1]Realtek Semiconductor Corp., Science-Based Industrial Park, Hsinchu, Taiwan

[2]Taiwan Semiconductor Manufacturing Company, Science-Based Industrial Park, Hsinchu, Taiwan

[3]Graduate Institute of Electronics Engineering & Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan

Abstract

Architecture and CAD are closely related issues in FPGA design. Routing architecture design shall optimize routability and facilitate router development; on the other hand, router design shall consider the specific properties of routing architectures to optimize the performance of the router. In this paper, we propose effective and efficient unified matching-based algorithms for array-based FPGA routing and segmentation design. For the segmentation design, we consider the similarity of input routing instances and formulate a net-matching problem to construct the optimal segmentation architecture. For the router design, we present a matching-based timing-driven routing algorithm which can consider a versatile set of routing segments. Experimental results show that our designed segmentations significantly outperform those used in commercially available FPGAs. For example, our designed segmentations achieve, on average, 14.6% and 19.7% improvements in routability, compared with those used in the Lucent Technologies ORCA 2C-series and the Xilinx XC4000E-series FPGAs, respectively.

1 Introduction

Due to their low prototyping cost, user programmability and short time-to-market, *Field Programmable Gate Arrays* (*FPGAs*) have become a popular choice for ASIC designs. Routing architectures and algorithms play a pivotal role in the implementation of circuits on FPGAs, especially for the large-scale FPGAs in the very deep submicron era [23]. A well-designed architecture could not attain its best performance without a corresponding router that can consider the specific properties of the routing architecture. Therefore, the designs of architectures and CAD tools are inseparable. This paper focuses on the unified design of routing architectures and routing tools for array-based FPGAs.

1.1 FPGA Architecture

Figure 1 shows a multi-segmented array-based FPGA architecture [18, 27]. A multi-segmented array-based FPGA consists of two-dimensional *logic modules* and routing resources (see Figure 1(a)). The logic modules, denoted by *L*, are customized for implementing various logic functions. The logic modules can be connected with the routing resources. The routing resources comprise vertical and horizontal routing channels and their intersection areas. A routing channel is composed of a set of tracks with segments of different lengths. (See Figure 1(b) for an example.) The area which a vertical and a horizontal channels intersect is referred to as a *switch module*, denoted by *S*. Each side of a switch module is linked with a set of segments. Segments on different sides of a switch module can be joined together through the switch module to form a longer connection.

1.2 Previous Work

Segmentation designs for FPGAs have been studied to some degree in the literature [3, 6, 11, 19, 22, 24, 29]. Most of the previous work either only deals with the segmentation design for row-based FPGAs ($i.e.$, channel segmentation design), or just points out their ideas for array-based FPGA segmentation design without providing specific algorithms or implementation. El Gamal *et al.* showed that a segmented routing channel can achieve comparable routability through appropriate arrangement of segment lengths. Roy and Mehendale in [24] and Zhu and Wong in [29] used stochastic methods for channel segmentation design. Pedram *et al.* in [22] later presented an analytical model for the design and analysis of segmented channel architectures.

Not much work has been reported for the segmentation design for array-based FPGAs. Zhu *et al.* observed that segmentation design for an array-based FPGA can be done in two stages—a segmentation design followed by a switch-module construction [28]. Based on the similar idea, Mak employed a decomposition procedure and showed how the two-stage approach can be done [19]. Chang *et al.* presented a matching-based algorithm for channel segmentation design and pointed out that the similar idea could be extended to the design for array-based FPGAs [6]. Betz and Rose explored the best routing architectures for multi-segmented array-based FPGAs with the considerations of buffer insertion and switch types [3].

Many routing algorithms for array-based FPGAs have been reported in the literature, e.g., [1, 2, 4, 5, 7, 9, 10, 14, 15, 16, 20, 25]. Most of these algorithms either consider routability only, without considering timing constraints [5, 7, 20], or just deal with one type of segments [1, 5, 7, 14, 16, 25]. Research on timing- and routability-driven routing for array-based FPGAs with segments of multiple lengths is still limited. Brown et al. in [15] proposed the SEGA detailed router which can deal with routability, timing, and multi-length segments at the same time. Based on the Pathfinder negotiated congestion algorithm (a maze router variant) [4, 9], Betz and Rose developed a combined placer and router called VPR [2] which considers only single-length lines. Betz et al. in [4] extended it to a combined placer and router that can consider more sophisticated area and timing models. A hierarchical timing-driven router for multi-segmented array-based FPGAs was presented in [8] by Chang et al.. The work intends to minimize the number of connections that violate timing constraints under area constraints.

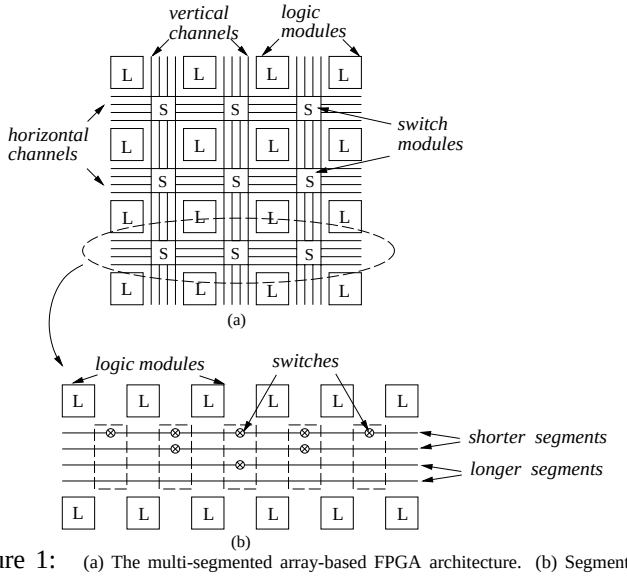

Figure 1: (a) The multi-segmented array-based FPGA architecture. (b) Segments of different lengths in a channel.

1.3 Our Contributions

In this paper, we present unified matching-based algorithms for multi-segmented array-based FPGA routing and segmentation design. We extend the approach for the channel segmentation design used in [6] to the array-based segmentation design. We first consider a *net-matching problem* and describe a matching-based algorithm to optimally solve the problem. Using the solution to the problem as a subroutine, we develop an effective matching-based algorithm for array-based FPGA segmentation design. For the multi-segmented array-based FPGA timing-driven routing, we propose a matching-based routing algorithm that can take advantage of the special properties of segmented routing architecture to optimize routablility, timing, or both. Unlike most existing algorithms that route net by net [1, 5, 7, 10, 14, 20, 26], our matching-based method, in particular, routes *a set of nets at a time*. Thus, with this more global perspective, our router can utilize various segments more effectively. We first consider delay-bound distribution and redistribution for each net, then formulate a subnet allocation problem for effectively utilizing wire segments for routing under timing constraints, and develop an efficient matching-based approach to solve the problem. Experimental results show that the segmentations constructed by our algorithm significantly outperform those used in commercially available FPGAs. For example, our designed segmentations, on average, achieve 14.6% and 19.7% improvements in routability, compared with those used in the Lucent Technologies ORCA 2C-series and the Xilinx XC4000E-series FPGAs, respectively.

The remainder of this paper is organized as follows. Section 2 formulates the segmentation design and segmented routing problem. Section 3 presents our matching-based algorithm for segmentation design. Section 4 proposes our matching-based approach for timing-driven routing. Finally, experimental results are reported in Section 5.

2 Problem Formulation

For an $N_v \times N_h$ (number of logic modules) array-based segmentation, there are $N_v - 1$ horizontal and $N_h - 1$ vertical routing channels. The channels in a

*The work of Yao-Wen Chang was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC-89-2215-E-009-054.

multi-segmented array-based FPGA are labeled $1, 2, ..., N_v - 1$ from the top to the bottom, and $N_v, N_v + 1, ..., N_v + N_h - 2$ from the left to the right. The length of a channel is measured by the number of columns. We number the columns $1, 2, \ldots, N_h + 1$ $(1, 2, \ldots, N_v + 1)$ from the left (or top) to the right (or bottom). A net I is divided into several *subnets*, namely i_x, $1 \leq x \leq q$, where q is the number of subnets in the net I. Let h_{i_x} denote the channel associated with the subnet i_x. A subnet i_x of net I in channel h_{i_x} is represented by a three-tuple $[source_{i_x}, target_{i_x}, h_{i_x}]$, where $source_{i_x}$ and $target_{i_x}$ are the leftmost (or topmost) and the rightmost (or bottommost) points of the subnets in a horizontal (or vertical) channel. Figure 2(a) shows a 3×3 array-based FPGA which contains four channels. There exists a net (global route) I which is composed of six subnets, namely $i_1, i_2, \ldots, i_6$. The subnet i_4 which ranges from column 2 to column 3 in channel 2 is represented as $[2, 3, 2]$ (see Figure 2(b)), and the net I is denoted by $\{[1, 2, 1], [2, 3, 3], [3, 4, 3], [2, 3, 2], [2, 3, 4], [3, 4, 1]\}$ (see Figure 2(a)).

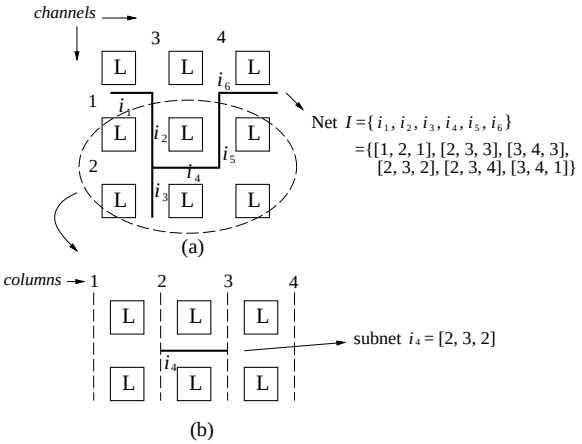

Figure 2: Representation of a net I in an array-based FPGA.

The *span* of subnet i_x is from $source_{i_x}$ to $target_{i_x}$. One subnet *overlaps* another if two subnets have the same channel number and the spans of the two subnets intersect. One net overlaps another if there exits a subnet in the net overlaps a subnet in the other. A segment *covers* a subnet if the span of the subnet is within the bounds of the segment, and the segment and the subnet are in the same channel. A set S of segments covers a routing instance R (i.e., a set of nets) if for each net I in R, there exists a set of consecutive segments in S that cover the net and no segment in the set covers more than one net. Let K be the timing constraint for a net, determined by the maximum number of segments allowed for the critical path in a net. For the *K-segment routing*, each routing path of a net can use up to K segments.

We formulate two closely related problems: one for segmentation design, and one for router design. *The Segmentation Design* and *the Timing-Driven Routing Problems for Multi-segmented Array-Based FPGAs* are formulated as follows:

- **The Segmentation Design Problem:** Given a set of routing instances (R_x, $1 \leq x \leq m$), a channel width N_t, and a timing constraint K (the maximum number of segments used for the critical path), design an array-based segmentation to maximize the success rate for K-segment routing.
- **The Timing-Driven Routing Problem:** Given a multi-segmented array-based FPGA, a netlist of global routes, and a timing constraint K (the maximum number of segments used for the critical path), determine the detailed routing paths for each net so that the routability is optimized and their timing constraints are satisfied.

3 Matching-Based Segmentation Design

We first formulate a net matching problem to find a *most economical* set S of segments which can cover each of given two input routing instances. Then, we apply a weighted bipartite matching algorithm to optimally solve the problem in polynomial time. Using the solution to the problem as a subroutine, we then develop an effective bottom-up matching-based algorithm for array-based FPGA segmentation design for an arbitrary number of input routing instances.

3.1 The Net Matching Problem

To formulate the net matching problem, we need to consider the spans of nets in a routing channel to quantify the "similarity" (overlap length) between two nets. Intuitively, it is more economical to design specific segments for nets with greater similarity. Since a net may contain multiple subnets in *one* channel, we shall first quantify the similarity between two subnets, then between two *sequences* of subnets *in a channel*, then between two nets, and finally between two sets of nets.

Let I and J be two overlapping *nets*. The overlap length between nets I and J, denoted by $olen(I, J)$. If nets I and J are merged, the resulting net, defined as $Nmerge(I, J)$. Let $\mathcal{I}$ and $\mathcal{J}$ be two finite *sets of nets*. A *net matching* M between $\mathcal{I}$ and $\mathcal{J}$ is a set of ordered pairs of intersecting nets $(I_1, J_1), (I_2, J_2), \ldots, (I_g, J_g)$, where $\mathcal{A}= \{I_1, I_2, \ldots, I_g\}$ and $\mathcal{B}= \{J_1, J_2, \ldots, J_g\}$ are two sets of distinct nets from $\mathcal{I}$ and $\mathcal{J}$, respectively. We can replace I_1 and J_1 by $Nmerge(I_1, J_1)$, replace I_2 and J_2 by $Nmerge(I_2, J_2)$, ..., and replace I_g and J_g by $Nmerge(I_g, J_g)$. After the replacement, the set of nets $\mathcal{I} \cup \mathcal{J}$ is represented as follows:

$$Union(\mathcal{I}, \mathcal{J}) = \{Nmerge(I_1, J_1), Nmerge(I_2, J_2), \ldots, Nmerge(I_g, J_g)\} \cup (\mathcal{I} - \mathcal{A}) \cup (\mathcal{J} - \mathcal{B}). \quad (1)$$

Let Length($\mathcal{I}$) be the total length of all nets in set $\mathcal{I}$. We formulate *the Net Matching Problem* described as follows:

- **The Net Matching Problem:** Given two finite sets $\mathcal{I}$ and $\mathcal{J}$ of nets, find a matching M such that $Length(Union(\mathcal{I}, \mathcal{J}))$ is minimized.

To solve the Net Matching Problem, we reduce the problem to computing the maximum matching in a weighted bipartite graph. Based on the weighted bipartite matching theory, we can optimally solve the Net Matching Problem in polynomial time. Given two finite sets $\mathcal{I}$ and $\mathcal{J}$ of nets, a weighted bipartite graph $G = (U, V, E)$ can be constructed. For each net I in $\mathcal{I}$ (J in $\mathcal{J}$), we introduce a vertex u_I (v_J) in the set U (V) of vertices. For each pair of overlapping nets, I, J, $I \in \mathcal{I}$ and $J \in \mathcal{J}$, connect u_I to v_J by an edge $e_{IJ} = (u_I, v_J) \in E$ with a weight computed by the weight function $\Psi : E \rightarrow Z^+$ defined as follows:

$$\Psi(e_{IJ}) = olen(I, J). \quad (2)$$

Then we can apply a maximum weighted bipartite matching algorithm [21] on G to solve the Net Matching Problem optimally.

A matching M of a graph $G = (U, V, E)$ is a subset of the edges with the property that no two edges of M share the same vertex. Edges in M are called *matched edges*; they are *unmatched*, otherwise. Let $Matched(\mathcal{I}, \mathcal{J})$ be the set of the matched edges in a weighted bipartite graph induced by the finite sets $\mathcal{I}$ and $\mathcal{J}$ of nets, and $Weight(Matched(\mathcal{I}, \mathcal{J}))$, $Matched(\mathcal{I}, \mathcal{J}) \subseteq E$, denotes the total weight of the edges in $Matched(\mathcal{I}, \mathcal{J})$. We have the following lemma and theorem.

Lemma 1 $Length(Union(\mathcal{I}, \mathcal{J})) = Length(\mathcal{I}) + Length(\mathcal{J}) - Weight(Matched(\mathcal{I}, \mathcal{J}))$.

Theorem 1 *The maximum weighted bipartite matching algorithm optimally solves the Net Matching Problem in $O((p_1 + p_2)^3 + q_1 q_2)$ time, where p_1 and p_2 (q_1 and q_2) are the numbers of nets (subnets) in the two input sets.*

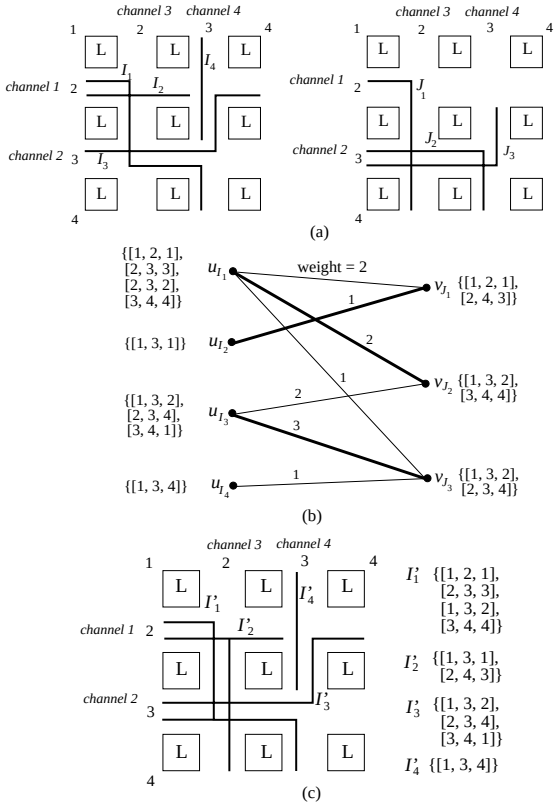

Figure 3: A matching-and-merging example for nets. (a) Two sets of nets. (b) The corresponding weighted bipartite graph. (c) The matching result for the two sets of nets.

Example 1 *Figure 3(a) shows two sets $\mathcal{I} = \{I_1, I_2, I_3, I_4\}$ and $J = \{J_1, J_2, J_3\}$ of nets in a multi-segmented array-based FPGA.The induced*

weighted bipartite graph is given in Figure 3(b). The spans of subnets of a net is shown next to its corresponding vertex. (Note that in this example, each net contains only one subnet in a channel.) The weight for each edge is computed by the function Ψ and shown beside the edge. The maximum weighted bipartite matching M between $U = \{u_1, u_2, u_3, u_4\}$ and $V = \{v_1, v_2, v_3\}$ is illustrated in Figure 3(b) by heavy lines. In this example, $M = \{(u_1, v_2), (u_2, v_1), (u_3, v_3)\}$. Note that u_4 is unmatched. Figure 3(c) shows the resulting configuration of replacing I_1 and J_2 by $Nmerge(I_1, J_2)$, I_2 and J_1 by $Nmerge(I_2, J_1)$, and I_3 and J_3 by $Nmerge(I_3, J_3)$. Let $I'_1 = Nmerge(I_1, J_2)$, $I'_2 = Nmerge(I_2, J_1)$, $I'_3 = Nmerge(I_3, J_3)$, and $I'_4 = I_4$. After the replacement, the set of nets $\mathcal{I} \cup \mathcal{J}$ becomes $Union(\mathcal{I}, \mathcal{J}) = \{I'_1, I'_2, I'_3, I'_4\}$. The reader can verify that $Length(Union(\mathcal{I}, \mathcal{J})) = len(I'_1) + len(I'_2) + len(I'_3) + len(I'_4) = 5 + 4 + 4 + 2 = 15$ is the minimum possible total union length for merging $\mathcal{I}$ and $\mathcal{J}$. (Note that $Length(Union(\mathcal{I}, \mathcal{J})) = Length(\mathcal{I}) + Length(\mathcal{J}) - Weight(Matched(\mathcal{I}, \mathcal{J})) = (4+2+2+4)+(3+3+3)-(2+1+3) = 15$.)

3.2 The Matching-Based Segmentation Design Algorithm

Similar to the work in [6], our design algorithm consists of three stages: (1) the matching-and-merging stage, (2) the tuning stage, and (3) the filling stage. In the matching-and-merging stage, we repeatedly apply the aforementioned weighted bipartite matching algorithm to merge input routing instances and generate a set $\mathcal{I}$ of nets that can cover each of the input routing instances. In the tuning stage, we find a set $\mathcal{I}'$ of nets from $\mathcal{I}$, $\mathcal{I}' \subseteq \mathcal{I}$, which can be packed (routed) into channels. In the filling stage, we fill the empty space between each pair of subnets and determine the switch locations in the N_t tracks of each channel to form a set of segments.

The matching-and-merging stage proceeds in a tree-like bottom-up manner. Given m routing instances $R_1, R_2, \ldots, R_m$, each with $p_1, p_2, \ldots, p_m$ nets, respectively. We apply the aforementioned weighted bipartite matching algorithm to merge R_1 and R_2, R_3 and R_4, R_5 and R_6, ... The process proceeds level by level in a bottom-up manner until a final merged routing instance is obtained. Then, subnets belonging to the same channel are extracted for further process.

Let $\mathcal{I}_F$ be the set of the nets in the final merged routing instance. We have the following theorem.

Theorem 2 *$\mathcal{I}_F$ covers $R_x, \forall x, 1 \leq x \leq m$.*

By Theorem 2, using a set S of segments covering $\mathcal{I}_F$ for 1-segment routing can route all routing instances $R_1, R_2, \ldots, R_m$. As mentioned earlier, however, there is usually a limitation on the number of tracks N_t in a routing channel. Therefore, it is not always possible to construct channels formed by all the segments in S.

In the tuning and the filling stages, we construct a segmentation architecture from the final merged routing instance $\mathcal{I}_F$ channel by channel. First, we apply the basic left-edge algorithm [17] to route the subnets in each channel. In the tuning stage, we sort the resulting tracks in the non-increasing order of their total lengths occupied by the subnets. The first N_t tracks are chosen for further construction, and the tuning stage is done. In the filling stage, we determine the switch locations on the tracks and fill the empty space between each pair of subnets in each track to form a set of segments. We place a horizontal switch on the position that makes the two resulting segments most balanced in length, if there is an empty space between two subnets on a track.

By Theorem 3, the time complexity of our algorithm is given by $O(m^3p^3 + m^2q^2)$, where m is the number of input routing instances and p (q) is the maximum number of nets (subnets) in an input routing instance. Note that empirically the number of resulting nets per routing instance grows only linearly as the matching-and-merging process proceeds, instead of exponentially in $\lceil \lg m \rceil$ steps $(p, 2p, 4p, \ldots, 2^{\lceil \lg m \rceil}p)$ as shown in the theoretic analysis.

Theorem 3 *Algorithm Seg_Designer runs in $O(m^3p^3 + m^2q^2)$ time, where m is the number of input routing instances and p (q) is the maximum number of nets (subnets) in a routing instance.*

4 Matching-Based Timing-Driven Routing

The goal of timing-driven routing is to route all nets in a circuit under the timing constraint. To perform timing-driven routing, we initially determine the delay bound for each subnet so that the timing constraint of each net is satisfied. Then, we formulate a subnet allocation problem to route subnets on wire segments; this problem is solved in polynomial time by a weighted bipartite matching algorithm. By repeatedly applying the solution to the problem as a subroutine, we present an effective matching-based algorithm for the segmented routing problem. After a subnet is routed, its remaining delay bound may be redistributed to increasing routing fexibility of other subnets. Thus, we also present a delay-bound redistribution algorithm to optimize the routing.

4.1 Delay-Bound Distribution

In this subsection, we shall first discuss the delay bound allocation for each subnet and then for each net. We use a tree to represent a net (global route) I (could be a two- or multi-terminal net). For each terminal or each junction of subnets, we introduce a vertex v. For each subnet i_x of net I, we introduce an edge e_x. The delay bound of the edge e_x (i.e., the maximum number of segments allowed to be used by a subnet) is denoted by d_{e_x}. We will use i_x and e_x interchangeably in the rest of the paper.

We need the following notations to explain our procedures:

- T: A routing tree representing a net.
- r: Root of thhe tree T.
- l: A longest path rooted at the root of a tree.
- n_l: Number of subnets on a path l; i.e., the path length of l.
- a_l (b_l): Source (Sink) of l (i.e., $a_l = r$).
- D_v: Delay bound for the tree rooted at vertex v.

Since the delay bound for a tree is K ($D_r = K$), the delay bound of the longest path l of the tree is also K.

The delay bound d_e for the edge e on a path l can be computed by Algorithm Path_Delay_Bound_Distribution. We distribute the delay bound d_e of edge e on l proportional to the ratio of the edge length to the total length of l. After edge delay bound for each edge on the longest path is determined, we should compute the delay bounds for trees rooted at the vertices on the longest path, and then record them at the vertices (denoted by D_v). The delay bound D_v of a vertex v on l is the sum of the edge delay bounds from v to b_l, or equivalently the difference between the delay bound of the longest path and the sum of the edge delay bounds from a_l to v, *i.e.,*

$$D_v = \sum_{e \in P(v, b_l)} d_e = D_r - \sum_{e \in P(a_l, v)} d_e,$$

where $P(x, y)$ is the path from vertex x to vertex y. More specifically, we have the following lemma.

Lemma 2 $D_v = \sum_{e \in P(v, b_l)} d_e \geq n_{P(v, b_l)}$ *for each v on the path l.*

Algorithm Net_Delay_Bound_Distribution for computing the delay bounds for the subnets of *net* I is described as follows. We first construct a tree T for the net I as described earlier. The root r of T is extracted and assigned the delay bound K—the timing constraint of net I. Then, for all paths from the root to sinks in T, number of subnets on the paths are computed and sorted in non-increasing order. (If two paths have the same number of subnets, they are sorted according to their lengths.) We repeat the following process to compute the delay bounds for all subnets. The first path l is extracted, and the delay bounds for all subnets on l is then computed by Algorithm Path_Delay_Bound_Distribution. After edge delay bounds on l are determined, we can compute the delay bound for each path rooted at a vertex on the l by Equation (3). The processing for the path l is then finished and removed from T, and then we repeat the same process on the resulting tree. By careful analysis, we have the following time complexity.

Theorem 4 *Algorithm Net_Delay_Bound_Distribution runs in $O(q \lg q)$ time for a net I, where q is the number of subnets in I.*

4.2 The Subnet Allocation Problem

A *clique* is a set of subnets in a channel, and all the subnets in the clique overlap each other. A *maximal clique* C is a clique in which there exists no subnet i_x such that $i_x \notin C$ and $C \cup \{i_x\}$ forms a new clique. For easier presentation, we use i_x and i_y to denote two subnets of *different* nets throughout the rest of this paper. A *maximum clique* is a maximal clique C with the maximum cardinality. Our approach is based on the following observation. If one subnet overlaps another, the two subnets cannot be assigned to the same track. Therefore, subnets in a clique must be allocated to different tracks.

Before detailing the subnet allocation process, we shall first define the *utilization-timing ratio*, $U_r(i_x, t)$, for a subnet routing on a set of consecutive segments in the track t of channel h_{i_x}. $U_r(i_x, t)$ is used to measure the utilization of occupied segments in the track t by the subnet i_x as follows:

$$U_r(i_x, t) = \frac{len(i_x)}{\sum_{1 \leq y \leq k} len(s_y)} + \frac{\alpha}{k}, \tag{3}$$

where $len(i_x)$ and $len(s_y)$ are the respective lengths of the subnet i_x and the segment s_y, where s_y, $1 \leq y \leq k$, is a segment occupied by the subnet i_x in the track t of channel h_{i_x}, k is the number of segments used, and α is a user's specified parameter. If routability is the only concern, α is set to zero and $U_r(i_x, t)$ becomes a pure segmentation utilization ratio. On the other hand, α is set to a positive real number for simultaneous routability and timing optimization. The subnet i_x is routable on the track t if all segments s_y, $1 \leq y \leq k$, in the track t are not used by other net; it is unroutable, otherwise.

A *subnet allocation* M between a clique C of subnets and a set H of tracks in a channel is a set of ordered pairs of routing results (i_1, t_1), (i_2, t_2), ..., (i_l, t_l), where $A = \{i_1, i_2, \ldots, i_l\}$ and $B = \{t_1, t_2, \ldots, t_l\}$ are a set of subnets and a set of tracks from C and H, respectively. The total ratio of subnets in C routed on the tracks in H, defined as $Ratio(C, H)$, is computed as follows:

$$Ratio(C, H) = \sum_{1 \leq x \leq l} U_r(i_x, t_x). \tag{4}$$

We define the *Subnet Allocation Problem* as follows:

- **The Subnet Allocation Problem:** Given a clique C of subnets and a set H of tracks in a channel, find an allocation M such that $Ratio(C, H)$ is maximized.

We can solve the Subnet Allocation Problem in polynomial time based on a weighted bipartite matching algorithm. Given two finite sets C of subnets and H of tracks in a channel, we construct a weighted bipartite graph $G = (U, V, E)$ as follows. For each subnet i_x in C and each track t in H, we introduce a vertex u_{i_x} (v_t) in the set U (V) of vertices. If a subnet i_x is routable on a track t ($i_x \in C$ and $t \in H$), connect u_{i_x} to v_t by an edge $e_{i_x,t} = (u_{i_x}, v_t) \in E$ with a weight computed by the weight function $\Phi : E \rightarrow \Re^+$ defined as follows:

$$\Phi(e_{i_x,t}) = U_r(i_x, t). \tag{5}$$

A *matching* M of a graph $G = (U, V, E)$ is a subset of the edges with the property that no two edges of M share the same vertex. Edges in M are called *matched edges*; they are *unmatched*, otherwise. For each edge $e_{i_x,t} \in E$, we would route the subnet i_x on the track t, and thus the segments occupied by i_x in the track cannot be used by other nets. If there exists a subnet in C which cannot match any track in T, then the routing fails in the given number of tracks. We have the following theorem:

Theorem 5 *The maximum weighted bipartite matching algorithm optimally solves the Subnet Allocation Problem in $O((|C| + |H|)^3)$ time, where C is the maximum clique of subnets and H is the set of tracks in a channel.*

4.3 Delay-Bound Redistribution

A subnet is *relaxable* if its delay bound is greater than the number of segments consumed by the subnet. To systematically reallocate the remaining delay bound of a relaxable subnet to unrouted subnets to increase routing flexibility without violating the timing constraint of a net, we propose the Delay_Bound_Distribution algorithm.

4.4 The Matching-Based Timing-Driven Routing Algorithm

Our timing-driven routing algorithm consists of two stages: (1) the distributing stage and (2) the allocating-and-redistributing stage. In the distributing stage, we determine the delay bound for each subnet to satisfy the timing constraint. In the allocating-and-redistributing stage, we repeatedly find a maximum clique and apply the the aforementioned weighted bipartite matching alogrithm to route subnets to tracks; the delay bounds of unrouted subnets may be redistributed after a maximum clique (a set of subnets) is routed into the track.

Given a routing instance with p nets, we determine the delay bounds for all subnets by applying Algorithm Net_Delay_Bound_Distribution described in Subsection 4.1. Then, subnets belonging to the same channel are extracted for further processing in the next stage. We shall process "critical" channels first. Therefore, those nets in the channel with higher density should be routed first than those in the channel with lower density. We compute the densities $Den[N_c]$'s of all channels to determine the routing ordering of channels. In each iteration of the while loop, an unprocessed channel with the highest density is extracted, and we repeatedly extract the maximum clique C from the remaining subnets in the channel and find the maximum matching between the subnets in C and the tracks in H by applying the aforementioned weighted bipartite matching algorithm. After all subnets in C are routed into the tracks, we should redistribute the remaining delay bounds to unrouted subnets for those relaxable subnets.

Note that finding the maximum clique in a given interval graph (which corresponds to the subnets in an FPGA routing channel) can be solved in $O(q \lg q)$ time, where q is the number of intervals (subnets) in the graph [12]. Also, the number of subnets in a clique cannot exceed the number of tracks in a routing channel. Therefore, the aforementioned weighted bipartite matching can be solved in $O(N_t^3)$ time, where N_t^3 is the number of tracks in a channel. We have the following theorem.

Theorem 6 *Algorithm Matching_Based_Router runs in $O({N_t}^3 L N_c)$ time, where N_t is the number of tracks in a channel, L is the length of channel, and N_c is the number of channels in a multi-segmented array-based FPGA.*

5 Experimental Results

We implemented our segmentation design and timing-driven routing algorithms in the C++ programming language on a PC with a Pentium II 266 microprocessor and 128 MB RAM. The weighted bipartite matching code was adopted from the public LEDA package. The routability of the segmentation architectures designed by our algorithm was compared with that of the Lucent Technologies ORCA 2C-series [18] and the Xilinx XC4000E-series [27] FPGAs, based on the timing-driven router presented in this paper.

Each routing channel in an ORCA 2C-series FPGA constains 8 single-length lines, 8 quad-length lines, and 4 longlines [18] (number of tracks $N = 20$), and that in an XC4000E-series FPGA contains 8 single-length lines, 4 double-length lines, and 6 longlines [27] (number of tracks $N = 18$). The single-length lines form a grid of horizontal and vertical lines that intersect at switch modules. The double-length (quad-length) lines consist of a grid of segments twice (quadruple) as long as the single-length lines. The longlines are a grid of segments that run the entire vertical or horizontal channel. Our experiments were based on the same numbers of tracks as those used by the two types of FPGAs.

To prepare for a suite of benchmark circuits for experimentation, we first generated pins on a 20×20 (number of logic modules; $L = 20$) FPGA, with their Manhattan distances based on the discrete, geometric, normal, and Poisson distribuitons, represented by "D", "Ge", "No", and "Po", respectively. We then routed each pin instance by the global router used in [7].

For the purpose of fair comparisons, we constructed two segmentation architectures, one with 18 tracks per channel ($N = 18$) and one with 20 tracks ($N = 20$) per channel for comparison with the respective XC4000E-series and ORCA 2C-series architectures, based on eight new sets of routing instances generated by the aforementioned procedure (each set with 100 routing instances and each instance with 400–700 routes). We then routed the 800 instances by the timing-driven router presented in this paper on our designed, the XC4000E-series, and the ORCA 2C-series architectures. Tables 1 and 2 list the comparisons for routing success rates between our designed and the XC4000E-series architectures and between our designed and the ORCA 2C-series ones, respectively. The results shows that our design, on average, outperforms the XC4000E-series (ORCA 2C-series) architecture by 14.8% and 22.6% (9.5% and 15.6%) improvements for 5- and 8-segment routing, respectively.

	$f(l)$	5-segment		8-segment	
		Ours	XC4000E	Ours	XC4000E
D_1	(0.9, 0.1, 0, 0, 0)	74	63	84	67
D_2	(0.8, 0.1, 0, 0, 0)	72	62	85	68
Ge_1	γ^l,γ = 0.1	72	64	82	68
Ge_2	γ^l,γ = 0.5	99	93	99	93
No_1	μ = 5, σ^2 = 4	78	66	89	69
No_2	μ = 3, σ^2 = 5	83	72	91	76
Po_1	$\lambda^l e^{-\lambda}/l!$, λ = 5	78	66	89	70
Po_2	$\lambda^l e^{-\lambda}/l!$, λ = 7	60	51	76	56
Average	-	77.0	67.1	86.9	70.9
Comparison	-	1.148	1.000	1.226	1.000

Table 1: Comparison between our designed architecture and the XC4000E-series one for the success rates for 5- and 8-segment routing ($L = 20, T = 18$).

	$f(l)$	5-segment		8-segment	
		Ours	ORCA 2C	Ours	ORCA 2C
D_1	(0.9, 0.1, 0, 0, 0)	75	67	86	73
D_2	(0.8, 0.1, 0, 0, 0)	72	66	87	74
Ge_1	γ^l,γ = 0.1	72	66	84	72
Ge_2	γ^l,γ = 0.5	99	94	99	94
No_1	μ = 5, σ^2 = 4	84	71	90	77
No_2	μ = 3, σ^2 = 5	79	76	92	80
Po_1	$\lambda^l e^{-\lambda}/l!$, λ = 5	79	71	90	77
Po_2	$\lambda^l e^{-\lambda}/l!$, λ = 7	60	55	78	64
Average	-	77.5	70.8	88.3	76.4
Comparison	-	1.095	1.000	1.156	1.000

Table 2: Comparison between our designed architecture and the ORCA 2C-series one for the success rates for 5- and 8-segment routing ($L = 20, T = 20$).

Both of our routing and segmentation design algorithms are quite efficient. The run time for routing an instance with 400–700 routes ranged from 1.9 sec to 3.8 sec, and those for designing a segmentation architecture ranged from 152 sec (Distribution: Ge_2; $N = 18$) to 415 sec (Distribution: No_1; $N = 20$).

6 Conclusion

In this paper, we have proposed unified matching-based algorithms for array-based FPGA segmentation and routing design. Experimental results have shown that the unified approaches are very efficient and lead to segmentation architectures with significantly higher routability than industrial ones. More importantly, our work provides an insight into the unified design of architecture and CAD for FPGAs.

References

[1] M. Alexander and G. Robins, "New Performance-Driven FPGA Routing Algorithm," *proc. of Design Automation Conf*., pp. 562-567, June 1995.

[2] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Researcher," *Int. Workshop on Field-Programmable Logic and Applications*, 1997, pp. 213–222.

[3] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density," *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*, pp. 59-68, Monterey, CA, Feb. 1999.

[4] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, February 1999.

[5] S. Brown, J. Rose, 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, J.-M Lin, and D.F. Wong, "A Matching-Based Algorithm for FPGA Segmentation Design," *IEEE Trans. CAD*, Vol. 20, No. 6, pp. 784–791, June 2001.

[7] Y.-W. Chang, S. Thakur, K. Zhu, and D.F. Wong, "A New Global Routing Algorithm for FPGAs," *Proc. of Int. Conf. on Computer-Aided Design*, pp 380-385, Nov. 1994.

[8] Y.-W. Chang, K. Zhu, and D.F. Wong, "Timing-driven Routing for Symmetrical-array-based FPGAs,"*ACM Trans. on Design Automation of Electronic Systems*, Vol. 5, no. 3, pp. 433–450, July 2001.

[9] C. Ebeling, L. McMurchie, S.A. Hauck and S. Burns, "Placement and Routing Tools for the Triptych FPGA", *IEEE Trans. on VLSI*, pp. 473–482, Dec. 1995.

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

[11] A. El Gamal, J. Greene, and V. Roychowdhury, "Segmented Channel Routing is Nearly as Efficient as Channel Routing (and Just as Hard)," *Advanced Research in VLSI*, pp. 193-221, 1991.

[12] F. Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph," *SIAM J. Computing*, Vol. 1, pp. 180-187, 1972.

[13] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, 1976.

[14] Y. -S Lee and C. -H Wu, "A Performance and Routability driven router for FPGA's considering path delay," *proc. of Design Automation Conf*., pp. 557-561, June 1995.

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

[16] G. G. Lemieux and 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.

[17] M. J. Lorenzetti and D. S. Baeder, Chapter 5: *Routing in Physical Design Automation of VLSI Systems*, Edited by B. Preas and M. Lorenzetti, Benjamin/Cummings, Menlo Park, CA, 1988.

[18] Lucent Technologies, *ORCA OR2CxxA (5.0 V) and OR2TxxA (3.3V) Series Field-Programmable Gate Arrays*, Allentown, PA, August 1996.

[19] W. K. Mak and D. F. Wong, "Channel Segmentation Design for Symmetrical FPGAs," *Proc. of IEEE Conf. on Computer Design*, Austin, TX, Oct 1997.

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

[21] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[22] M. Pedram, B.S Nobandegani, and B.T. Preas, "Design and Analysis of Segmented Routing Channels for Row-Based FPGAs," *IEEE Trans. on Computer Aided Design*, Vol. 13, No. 12, pp. 1470-1479, Dec. 1994.

[23] J. Rose and D. Hill, "Architectural and Physical Design Challenges for One-Million Gate FPGAs and Beyond," *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*, pp. 129-132, Monterey, CA, Feb. 1997.

[24] K. Roy and M. Mehendale,"Optimization of Channel Segmentation for Channelled Architecture FPGAs," *Proc. of IEEE Custom Integrated Circuits Conf*., pp. 4.4.1-4.4.4, Boston, MA, May 1992.

[25] Y.-L. Wu and M. Marek-Sadowska, "Orthorgonal Greedy Coupling — A New Optimization Approach to 2-D FPGA Routing," *Design Automation Conference*, June 1995.

[26] Y.-L. Wu, S. Tsukiyama, and M. Marek-Sadowska, "Graph based analysis of 2-D FPGA routing," *IEEE Trans. Computer-Aided Design*, Vol. 15, No. 1, pp. 33–44, 1996.

[27] Xilinx Inc., *The Programmable Logic Data Book*, San Jose, CA, 1996.

[28] K. Zhu, D. F. Wong, and Y.-W. Chang, "Switch Module Design with Application to Two-Dimensional Segmentation Design," in *Proc. of IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 480-485, Santa Clara, CA, 1993.

[29] K. Zhu and D. F. Wong, "Segmented Channel Segmentation Design for Row-Based FPGAs," in *Proc. of IEEE/ACM Int. Conf. on Computer Aided Design*, pp. 26-29, Santa Clara, CA, 1992.