Efficient Routability Check Algorithms for Segmented Channel Routing

CHENG-HSING YANG
Kung Shan Institute of Technology
SAO-JIE CHEN
National Taiwan University
JAN-MING HO
Academia Sinica
and
CHIA-CHUN TSAI
National Taipei University of Technology

The segmented channel-routing problem arises in the context of row-based field programmable gate arrays (FPGAs). Since the K-segment channel-routing problem is NP-complete for $K \geq 2$, an efficient algorithm using the weighted bipartite-matching approach is developed for this problem. Connections that form a maximum clique are chosen first to be routed to the segmented channel. Then, another maximum clique of the remained connections is routed until all connections have been processed. In addition, a powerful "unroutability check" algorithm is uniquely proposed to tell whether the horizontal switches in an interval of the segmented channel are sufficient for routing or not. Hence, we can precisely discriminate the routable and the unroutable ones from all the test cases. As shown in the experiments, average discrimination ratios of 98.8% and 99.4% are obtained for the 2-segmentation and 3-segmentation models, respectively. Moreover, when applying our routing algorithm to the analyzed nonunroutable cases, a routing failure ratio of 1.5% is reported for the 2-segmentation model, compared to Zhu and Wong's 5.9%; also, a routing failure ratio of 0.8% (less than their 4.7%) is obtained for the 3-segmentation model. In total, the routing failure ratio of our routing algorithm is less than 21% of Zhu and Wong's.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Gate arrays*; B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design* (CAD)

This work was supported by the National Science Council R.O.C under grant NSC 86-2221-E002-066 and NSC 86-2221-E027-003.

Authors' addresses: C.-H. Yang, Department of Information Management, Kung Shan Institute of Technology, Tainan, Taiwan; S.-J. Chen, Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan; J.-M. Ho, Institute of Information Science, Academia Sinica, Taipei, Taiwan; C.-C. Tsai, Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan.

General Terms: Algorithms, Design

Additional Key Words and Phrases: Field programmable gate arrays (FPGAs), routing, segmented channel

1. INTRODUCTION

The segmented-routing channels, or simply the segmented channels, in a row-based FPGA [El Gamal et al. 1989] consist of vertical and horizontal routing segments. Each input or output of the logic modules connects to a dedicated vertical segment; programmable switches are located at each crossing of the vertical and horizontal segments (*cross switches*) and also between pairs of adjacent horizontal segments on the same track (*horizontal switches*), such that the routing in a row-based FPGA can be performed by programming these switch elements [Greene et al. 1990; El Gamal et al. 1991]. Researches on how to design a segmented channel that maximizes the routability and satisfies performance requirements have been reported [Zhu and Wong 1992; Burman et al. 1992; Pedram et al. 1994], and this is denoted as a *segmentation design problem*.

A *K-segment channel routing* is a routing that assigns each connection to a track such that no segment is occupied by more than one connection and each connection occupies at most K segments. Greene et al. [1990] state that the K-segment ($K > 1$) channel-routing problem is equivalent to the problem of numerical matching with target sums [Garey and Johnson 1979], and hence strongly NP-complete for $K \geq 2$. Exhaustive search [Greene et al. 1990; Roychowdhury et al. 1993]; bounded search [Roy 1993]; and heuristic algorithms [Zhu and Wong 1992] have been proposed to solve the *K-segment channel-routing* problem.

In this paper we develop a weighted bipartite-matching algorithm for the segmented channel-routing problem. We do not choose and assign connections one by one but clique by clique. The set of connections forming a maximum clique is chosen first. The routing of each clique of connections is done by finding a minimum weighted matching. If some connections cannot be assigned by the above step, postprocessing is used to reroute these connections. Also, in order to pick out the cases that are unroutable, we propose some criteria to check the unroutability of a case. Generally, given a set of connections to be routed over a range of columns, the number of tracks must be large enough such that each of the overlapping connections can be assigned to a different track, and the number of switches must also be sufficient such that as many nonoverlapped connections as possible can be assigned to the same track. Accordingly, a simple but powerful unroutability check method is proposed to tell whether the tracks and the switches between a range of columns are sufficient to complete the connections in this range.

The remainder of this paper is organized as follows: Section 2 describes some preliminary concepts and basic definitions. The routing and postprocessing algorithms are presented in Section 3. The unroutability check algorithm is stated in Section 4. Results on a set of benchmarks are reported in Section 5. Finally, a conclusion is drawn in Section 6.

2. PRELIMINARIES AND DEFINITIONS

In this paper we refer to Roychowdhury [1993] for the following definitions. The input to a segmented channel-routing problem, as depicted in Figure 1, is a segmented channel consisting of a set $\mathbf{T}$ of T tracks and a set $\mathbf{C}$ of M connections. A symbol ○ represents a horizontal switch and ⊕ represents a cross switch. Each track extending from column 1 to column N is divided into a set of adjacent segments separated by horizontal switches.

For a segment s, $left(s)$ and $right(s)$ denote the leftmost and rightmost columns in which this segment is present. Similarly, a connection c is uniquely characterized by the span of its leftmost and rightmost columns, $left(c)$ and $right(c)$. When a connection c is *assigned* to a track t, the one or more segments in track t that are present in the columns spanned by c are considered *occupied*. More precisely, a segment s in track t is occupied by the connection c if $right(s) \geq left(c)$ and $left(s) \leq right(c)$. For a K-segment channel routing, c can be assigned to t only if c occupies at most K segments. A connection c is *assignable* (to a track t) if there exists such a track t to which c may be *assigned* and if the segments in track t that are present in the columns spanned by c are not *occupied* by another connection. Connections c_i and c_j *overlap* if their spans overlap. Connection c_i *collides* with c_j on a track t if, when they are assigned to t, some of their occupied segments will overlap. For the segmented channel-routing problem, we restrict each connection that may be assigned to a single track only. The main objective of a K-segment channel routing is to find a *full assignment* of connections to tracks, i.e., to complete the assignment of connections to tracks such that each of the connections occupies at most K segments. Another objective is to reduce the number of switches used in the routing.

Given a set $\mathbf{C}$ of M connections with density D, which is the maximum number of connections over a column, a *clique* is a subset of $\mathbf{C}$ such that any c_i and c_j in it overlap each other. A *maximal clique* $\mathbf{Q}$ is a clique, in which there is no $c_i \notin \mathbf{Q}$ such that $\{c_i\} \cup \mathbf{Q}$ forms a new clique. A *maximum clique* is a maximal clique $\mathbf{Q}$ with the maximum cardinality, i.e., $|\mathbf{Q}| = D$. Note that any two connections in a clique cannot be assigned to the same track for they will occupy some same segments. A clique and the set of tracks form the two sets of nodes of a bipartite graph G. To find a full assignment of the clique $\mathbf{Q}$ is to find a maximum matching of the bipartite graph such that the cardinality of the maximum matching is equal to $|\mathbf{Q}|$.

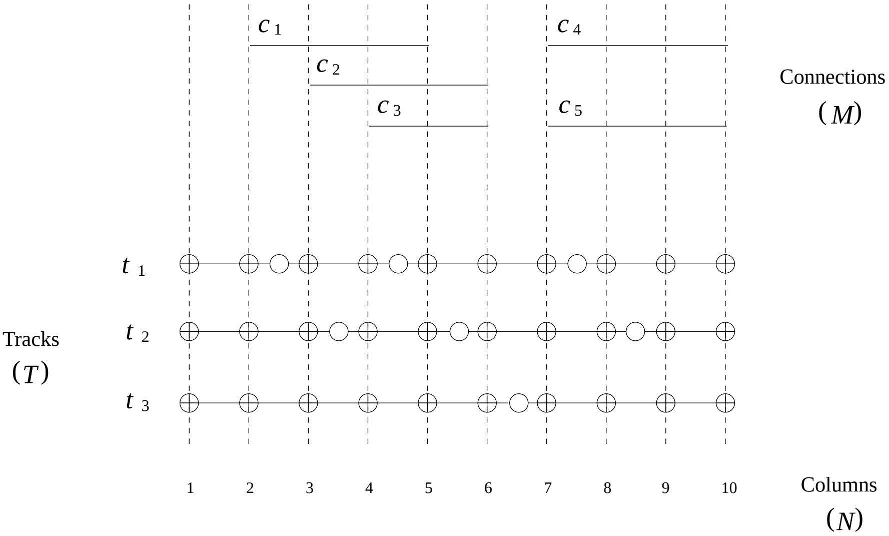

Fig. 1. The model of a segmented channel routing.

3. THE ROUTING ALGORITHM

To solve the segmented channel-routing problem efficiently, we apply the weighted bipartite matching algorithm to route and remove cliques of connections one by one. The weighted matching problem can be solved in time complexity $O(n^3)$ for a complete bipartite graph with $2n$ nodes [Papadimitriou and Steiglitz 1982]. The routing algorithm is as follows:

Algorithm. ***C_route***(**C**, **T**)

```
{  Do until C is empty:
   {  Q = Remove_Max_Clique(C);
      G = Construct_Bipartite_Graph(Q, T);
      (Routed, Unrouted) = Weighted_Bipartite_Matching (G);
      Total_Routed = Total_Routed ∪ Routed;
      Total_Unrouted = Total_Unrouted ∪ Unrouted;
   }
   if (Total_Unrouted is empty)
      return(feasible);
   else
      return(Postprocessing(Total_Unrouted));
}
```

Every time a maximum clique **Q** of the remaining connections is picked out, a weighted bipartite graph for **Q** is constructed. The weight of edge e_{ij} is defined as follows:

$$weight(e_{ij}) = \begin{cases} \alpha \cdot w_1 + \beta \cdot w_2 + \gamma \cdot w_3 & \text{if } c_i \text{ is assignable to } t_j \\ \infty & \text{otherwise,} \end{cases} \tag{1}$$

where

$$w_1 = \frac{collision\_number(c_i, t_j)}{remained\_connection(\mathbf{C})},$$

$$w_2 = \frac{segment\_length(c_i, t_j) - connection\_length(c_i)}{segment\_length(c_i, t_j)},$$

and

$$w_3 = \frac{switch\_number(c_i, t_j)}{K - 1}.$$

In the above equations, $collision\_number(c_i, t_j)$ is the number of remaining connections that collide with c_i on track t_j; $remained\_connection(\mathbf{C})$ is the total number of remaining connections in $\mathbf{C}$; $segment\_length(c_i, t_j)$ is the total length of segments occupied by c_i when c_i is assigned to t_j; and $connection\_length(c_i)$ is the length of c_i; $switch\_number(c_i, t_j)$ is the number of switches that need to be programmed when c_i is assigned to t_j. α, β, and γ are ratio parameters; w_1 denotes the degree of popularity on track t_j for the remaining connections; and w_2 and w_3 denote the *segment-length waste* and *switch usage*, respectively. In order to save resources of segments and switches, a connection should be assigned to a track such that segment-length waste and switch usage are lower. Also, a connection should be assigned to a track with lower cost of w_1 such that fewer remaining connections will be affected. *Weighted_Bipartite_Matching* finds a minimum weighted match. If connection c_i is matched with track t_j and $weight(e_{ij}) \neq \infty$, c_i can be assigned to t_j and is put into the *Routed* set. Otherwise, c_i cannot be assigned and has to be put into the *Unrouted* set. The *Total_Unrouted* set is used to save all the connections not yet assigned. If *Total_Unrouted* is empty, this case is routable. Otherwise, a *Postprocessing* procedure is called to reroute the connections in *Total_Unrouted* again.

Postprocessing

When the routing of some connections cannot be completed in the routing step, we have to reroute them one-by-one using the *Postprocessing* step. But first we need some definitions. If a connection has been assigned to a track but is also assignable to another track, we say this connection is *movable*. When a connection c_i is rerouted, it has to *squeeze* one connection c_j assigned to a track t; that is, we have to remove c_j from t when assigning c_i to t. Of course, this would only be done if the removal of c_j implies that c_i becomes assignable to t. If c_j is *movable*, the rerouting of c_i is done. Otherwise, c_j has to squeeze another connection again. We do not consider the case where a connection would squeeze more than one connection on a track at a time, because this is too complex. The idea of squeezing can be implemented by the following algorithm.

***Algorithm*. Postprocessing(Total_Unrouted)**

```
{   Do until $Total\_Unrouted$ is empty:
    {   $c$ = remove a connection from $Total\_Unrouted$;
        $Queue = \emptyset$;
        Put all movable connections into $Queue$ and mark these connections as
        roots;
        Do while $Queue$ is not empty and $c$ is not yet in $Queue$:
        {   $c_q$ = remove a connection from $Queue$;
            $A$ = the connections that have never been in $Queue$ and can squeeze $c_q$;
            link all connections in $A$ to $c_q$;
            put all connections in $A$ into $Queue$;
        }
        if $c$ is not in $Queue$
            return(infeasible);
        else
            reassign the connections in the squeeze path for $c$;
    }
    return(feasible);
}
```

To reroute an unrouted connection c, all *movable* connections that become the roots of the squeeze paths are put into a *Queue* first. The connections, which have not been in *Queue* but can squeeze one connection c_q in *Queue*, are linked to c_q and put into *Queue*. If the unrouted connection c is in *Queue*, the squeezing path for c is found. Otherwise, c cannot be rerouted by this approach.

Time Complexity

Procedures *Construct_Bipartite_Graph*, *Weighted_Bipartite_Matching*, and *Postprocessing* are the main parts of Algorithm *C_route*. Let $q_1, q_2, \cdots, q_m$ be the cardinalities of cliques that are processed step by step in *C_route*. For each step, say q_k, each $weight(e_{ij})$ in *Construct_Bipartite_Graph* can be calculated at $O(M)$ time. There are $O(T^2)$ edges in a bipartite graph. Thus, *Construct_Bipartite_Graph* takes $O(MT^2)$ time. Also, *Weighted_Bipartite_Matching* takes $O(T^3)$ time. So to process m steps it takes $O(mMT^2 + mT^3)$ time, which is less than $O(M^2T^2 + MT^3)$.

Suppose n connections need to be rerouted in *Postprocessing*. Each connection takes $O(MT)$ time to find the movable connections and to construct a squeeze path. Hence, *Postprocessing* takes $O(nMT)$ time, which is less than $O(M^2T)$. Thus the time complexity for Algorithm *C_route* is $O(M^2T^2 + MT^3)$.

Critical Path

Our algorithm is designed for the K-segment channel routing problem. It does not consider the problem of critical path in a routing case. There are two possible methods of adjusting our algorithm to handle the problem of critical path. The first one is to route the connections in a critical path by first setting the γ value to be much larger than the α and β values. Then *C_route* is used to route the remaining connections if they are routable. The second method uses *C_route* to route all connections as before, but the weight of each edge created by the connections in a critical path is set as follows: its γ value must be larger than the α and β values.

4. UNROUTABILITY CHECK

In the *C_route* algorithm introduced in Section 3, if the returned signal in *C_route* is feasible then, of course, this case is routable. But if *C_route* cannot find a feasible routing, it is not sufficient to say that it is unroutable. In this section we present the sufficient conditions for an unroutable case and develop an algorithm to check whether the case is really unroutable.

To describe the following lemmas, we assume that a K-segment channel-routing problem includes a set $\mathbf{C}$ of M connections, a set $\mathbf{T}$ of T tracks, and the density of $\mathbf{C}$ is D. Let $\mathbf{Q}$ be a clique of $\mathbf{C}$ and G be the bipartite graph built from $\mathbf{Q}$ and $\mathbf{T}$. If the cardinality of the maximum matching for G is equal to $|\mathbf{Q}|$, $\mathbf{Q}$ can be completely assigned. We call such a G *fully matchable*.

LEMMA 1. *Given a clique* $\mathbf{Q}$ *of* $\mathbf{C}$*, if its bipartite graph* G *is not fully matchable,* $\mathbf{C}$ *is unroutable.*

PROOF. G not being fully matchable means that $\mathbf{Q}$ is unroutable. Because $\mathbf{Q} \subseteq \mathbf{C}$, $\mathbf{C}$ is unroutable. □

Let $x = left(c)$ and $\mathbf{Q}_x$ be the clique that consists of the connections spanning over column x, and $q_x = |\mathbf{Q}_x|$. As shown in Figure 2, when we scan the connections from column x to the left until a certain column x', which is the rightmost column of a certain connection c', we find k_x rightmost points appearing between $x - 1$ and x', and these k_x connections form a clique $\mathbf{Q}_{x'}$. Suppose there exists a set $\mathbf{T}_x$ of t_x tracks having switches between x and x', we then have the following lemma.

LEMMA 2. *If* $\mathbf{C}$ *is routable, then* $T - q_x + t_x \geq k_x$.

PROOF. First, if each track in T has no switch between $x - 1$ and x', any two connections chosen from $\mathbf{Q}_x$ and $\mathbf{Q}_{x'}$ will collide with each other on this track. In this case, if $|\mathbf{Q}_x| + |\mathbf{Q}_{x'}| > T$, then $\mathbf{C}$ is unroutable. Second, since a track in $\mathbf{T}_x$ allows at most two connections, one from $\mathbf{Q}_x$ and the

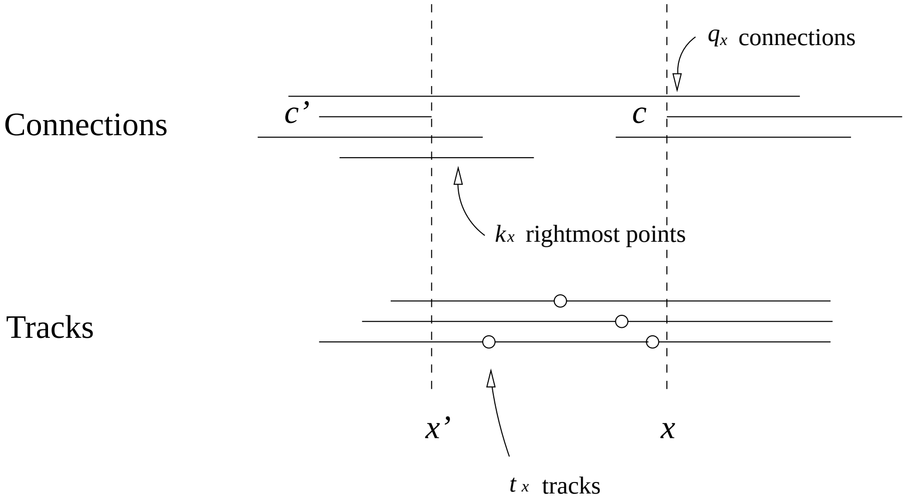

Fig. 2. Routability with the necessary condition $T - q_x + t_x \geq k_x$.

other from $\mathbf{Q}_{x'}$, to be assigned, if $|\mathbf{Q}_x| + |\mathbf{Q}_{x'}| - |\mathbf{T}_x| > T$, $\mathbf{C}$ is unroutable. Hence we have proven this lemma. □

More precisely, let m_x be the cardinality of the maximum matching for the bipartite graph built from k_x connections and t_x tracks. If $\mathbf{C}$ is routable, then $T - q_x + m_x \geq k_x$. Lemma 2 is also applicable to the case of column $x = right(c)$, and the difference is that it scans the connections from column x to the right and counts the connections whose leftmost points have been scanned.

Lemma 2 only considers the case where the k_x-counted connections form a clique. If some complete connections were to appear between $x - 1$ and x', it may be the case that two counted connections were assigned to the same track. To deal with this case, we have to change Lemma 2 by considering the following two conditions. Condition (1): there is no complete segment that appears between $x - 1$ and x'. Condition (2): there exist complete segments that appear between $x - 1$ and x'. Considering Condition (1), since all counted connections cannot be assigned to the same track in $\mathbf{T}_x$, Lemma 2 is still suitable. Considering Condition (2), we have to partition k_x connections into two parts: one consisting of k_{x_1} connections that do not appear complete between $x - 1$ and x'; the other consists of k_{x_2} connections that appear complete between $x - 1$ and x'. Let s_x be the number of segments that appear complete between $x - 1$ and x'. In order to check unroutability, we need to know that how many k_{x_2} connections can be assinged to the s_x segments. This problem is similar to the K-segment channel-routing problem, and thus is NP-complete. However, we can calculate the maximum number of connections to be assigned to each track, which is formed by these s_x segments independently. Then, we obtain an upper bound of the above problem by totalizing these maximum numbers.

The maximum number can be calculated easily if we sort in increasing order the k_{x_2} connections by their rightmost columns, and then assign them to a track according to the sorting order. Let the total be s'_x and

$$k'_{x_2} = \begin{cases} k_{x_2} - s'_x & \text{if } k_{x_2} > s'_x \\ 0 & \text{if } k_{x_2} \le s'_x. \end{cases} \quad (2)$$

Thus, there are at least $k_{x_1} + k'_{x_2}$ counted connections that cannot be assigned to the same track in $\mathbf{T}_x$. Note that Condition (1) is a special case of Condition (2) for $s_x = 0$. Now we have the following lemma:

LEMMA 3. *If* **C** *is routable, then* $T - q_x + t_x \ge k_{x_1} + k'_{x_2}$.

Given connections **C** and tracks **T**, the *Unroutability_check* algorithm uses Lemma 3 to check whether **C** is unroutable. The cardinalities of each clique of connections $\mathbf{Q}_x$ can be calculated column by column in linear time if connections in **C** are sorted [Rheinboldt 1980]. In order to save time, only the columns whose $|\mathbf{Q}_x|$ is larger than ($D -$ CLIQUE_CONSTANT) are processed. In our experiments, almost no unroutable cases are found when CLIQUE_CONSTANT is set to 8. Scanning from column x to the left, whenever a connection c' and its rightmost column x' is found, Lemma 3 is used to check the unroutability. If there exist x and x' such that $T - q_x + t_x < k_{x_1} + k'_{x_2}$, **C** is unroutable. Otherwise, Algorithm *Unroutability_check* fails to check whether **C** is unroutable.

***Algorithm*. Unroutability_check(C, T)**

```
{  To calculate $|\mathbf{Q}_x|$ of each column $x$ for C;
   For each column $x$ such that $|\mathbf{Q}_x| > (D -$ CLIQUE_CONSTANT$)$ do:
   {  if ($\mathbf{Q}_x$ is a maximal clique and $x$ is the leftmost column of a certain connection)
         scan from $x$ to the left to find $x'$ columns and use Lemma 3 to check unroutability;
      if ($\mathbf{Q}_x$ is a maximal clique and $x$ is the rightmost column of a certain connection)
         scan from $x$ to the right to find $x'$ columns and use Lemma 3 to check unroutability;
   }
   if (one of above checks does not satisfy Lemma 3)
      return(unroutable);
   else
      return(fail to check);
}
```

For each column x and maximal clique $\mathbf{Q}_x$, it takes $O(MT)$ time to assign k_{x_2} connections to each of t_x tracks after a column x' is found. Since there are at most $O(M)$ of maximal cliques and the number of x' columns that each maximal clique $\mathbf{Q}_x$ needs to scan is $O(M)$, the time complexity of Algorithm *Unroutability check* is $O(M^3T)$.

Table I. Routability Analysis Results for the K-Segmentation Models

	C_route				*Unroutability_check*				DR	
Segmented channel	$K = 2$		$K = 3$		$K = 2$		$K = 3$		$K = 2$	$K = 3$
	RC	sec.	RC	sec.	UC	sec.	UC	sec.		
D1	207	61.8	185	68.1	92	9.5	115	10.1	99.7%	100.0%
D2	258	54.2	257	60.9	42	3.5	43	3.6	100.0%	100.0%
D3	203	76.3	186	88.5	95	30.0	113	28.6	99.3%	99.7%
D4	211	98.1	203	119.9	80	74.6	94	67.6	97.0%	99.0%
D5	174	68.1	180	72.8	126	8.1	120	12.1	100.0%	100.0%
D6	250	49.6	211	60.6	50	10.2	89	7.2	100.0%	100.0%
D7	208	180.0	217	201.5	87	188.1	78	186.1	98.3%	98.3%
Geometric	164	169.5	165	188.1	133	82.7	135	83.9	99.0%	100.0%
Normal	267	47.8	292	62.6	23*	33.6	4	27.1	96.7%	98.7%
Poisson	250	140.1	250	152.6	45	174.0	44	167.0	98.3%	98.0%
Average	219	94.6	214	107.3	77	44.9	84	59.3	98.8%	99.4%

*This includes 7 unroutable cases which are found by Lemma 1.

5. EXPERIMENTAL RESULTS

Our *C_route* and *Unroutability_check* algorithms were implemented in the C language and tested on the 300 set connections created by Zhu and Wong [1992]. All experiments ran on a SUN Sparc 10 workstation with 32 MB main memory. The parameters of the channel model, $N = 100$ and $T = 36$, are close to the Actel's ACT2 family A1280 FPGA [Actel Corporation 1991]. The ratio parameters α, β, and γ in our experiments are 0.5, 0.4, and 0.1, respectively. Zhu and Wong's 2-segmentation models and 3-segmentation models were used; ten types of connection distributions, D1 through D7, geometric, normal, and Poisson, were tested. Table I shows the 2-segment and 3-segment channel-routability analysis results using *C_route* and *Unroutability_check* on each distribution of 300 test cases. RC is the number of cases that can feasibly be routed by *C_route*. UC is the number of cases that can be checked as unroutable by *Unroutability_check*. The total time to run the 300 cases for each distribution is also shown in the table. The last two columns show the discrimination ratios obtained from the results of these two algorithms, i.e., the ratio of RC + UC cases over 300 cases for each distribution. The discrimination ratio is denoted as DR. The average discrimination ratios are 98.8% and 99.4% for 2-segment and 3-segment channel routing, respectively.

Table II and Table III compare the 2-segment channel-routing and 3-segment channel-routing results, respectively, of Zhu and Wong's routing algorithm with our *C_route* algorithm. Column 2 shows the number of test cases, excluding the UC cases for each distribution. FC is the number of cases that fail to be routed. The failure ratios of our routing algorithm are all less than or equal to those of Zhu and Wong's. For the 2-segment channel routing, ours is less than 25% of theirs on the average. For the 3-segment channel routing, ours is less than 17% of theirs on the average. The disadvantage of our approach is that our runtime is longer than theirs.

Table II. Comparing Zhu and Wong's 2-Segment Channel Routing Algorithm with Others

Segmented channel	No. of cases	*Zhu and Wong*		*C_Route*		Failure ratio	
		FC	sec.	FC	sec.	*Zhu and Wong*	*C_route*
D1	208	8	4.5	1	33.4	3.8%	0.5%
D2	258	1	3.3	0	43.0	0.4%	0.0%
D3	205	9	9.0	2	35.2	4.4%	1.0%
D4	220	21	15.3	9	49.0	9.5%	4.1%
D5	174	5	4.4	0	26.5	2.9%	0.0%
D6	250	11	4.4	0	36.8	4.4%	0.0%
D7	213	30	23.0	5	77.0	14.1%	2.3%
Geometric	167	11	14.7	3	50.1	6.6%	1.8%
Normal	277	17	8.2	10	41.1	6.1%	3.6%
Poisson	255	17	24.4	5	91.6	6.7%	2.0%
Average	223	13	11.1	4	48.4	5.9%	1.5%

Table III. Comparing Zhu and Wong's Algorithm on 3-Segment Channel Routing

Segmented channel	No. of cases	*Zhu and Wong*		*C_Route*		Failure ratio	
		FC	sec.	FC	sec.	*Zhu and Wong*	*C_route*
D1	185	0	5.3	0	30.7	0.0%	0.0%
D2	257	0	4.5	0	47.3	0.0%	0.0%
D3	187	21	12.9	1	35.3	11.2%	0.5%
D4	206	7	17.7	3	50.9	3.4%	1.5%
D5	180	0	5.5	0	30.2	0.0%	0.0%
D6	211	10	5.2	0	34.9	4.7%	0.0%
D7	222	28	34.4	5	93.3	12.6%	2.3%
Geometric	165	12	20.4	4	50.4	7.3%	0.0%
Normal	296	9	10.6	4	60.7	3.0%	1.4%
Poisson	256	13	28.6	6	101.1	5.1%	2.3%
Average	217	10	14.5	2	53.5	4.7%	0.8%

The results of postprocessing are shown in Table IV. Columns 3 to 6 show the number of cases routed with the help of postprocessing for Zhu and Wong's and our algorithms. The ratios are the numbers of routed cases over the cases numbers shown in Table II and Table III. Although the benefit obtained by postprocessing is small, it is helpful when the failure ratio of the main routing algorithm is quite small also. This also means that our weighted bipartite matching approach is efficient.

We speculated that our routing algorithm would work badly if N were larger than the average length of the connections. So we tested our algorithm for the channel models using different N values. First, we used Zhu and Wong's program to generate 3-segmentation models for N = 100, 200, · · ·, 2000. The results for N = 2000 are shown in Table V. RC_1 and RC_2 are the numbers of cases which can feasibly be routed by, respectively, Zhu and Wong's algorithm and our *C_route*. UC is the number of cases that can be checked unroutable by *Unroutability_check*. DR is the discrimination ratio. DF is the difference for the failure ratio, which is the value of Zhu and Wong's failure ratio minus our failure ratio. Our *C_route* is still

Table IV. Results of Postprocessing on the K-Segmentation Models

Segmented channel	Routed by *postprocessing*				Ratio			
	Zhu and Wong		*C_route*		*Zhu and Wong*		*C_route*	
	$K = 2$	$K = 3$	$K = 2$	$K = 3$	$K = 2$	$K = 3$	$K = 2$	$K = 3$
D1	0	0	0	0	0.0%	0.0%	0.0%	0.0%
D2	0	0	0	0	0.0%	0.0%	0.0%	0.0%
D3	0	0	0	0	0.0%	0.0%	0.0%	0.0%
D4	1	1	8	1	0.5%	0.5%	3.6%	0.5%
D5	0	0	0	0	0.0%	0.0%	0.0%	0.0%
D6	0	0	0	0	0.0%	0.0%	0.0%	0.0%
D7	12	3	12	8	5.6%	1.4%	5.6%	3.6%
Geometric	0	0	2	4	0.0%	0.0%	1.2%	2.4%
Normal	3	1	6	0	1.1%	0.3%	2.2%	0.0%
Poisson	0	0	2	2	0.0%	0.0%	0.8%	0.8%
Average	1.6	0.5	3.0	1.5	0.7%	0.2%	1.3%	0.7%

Table V. Routability Analysis for the 3-Segmentation Models with N = 2000

Segmented channel	Average connection length	RC_1	RC_2	UC	DR	DF
D1	986	173	173	127	100.0%	0.0%
D2	1334	249	249	51	100.0%	0.0%
D3	646	140	152	147	99.7%	7.8%
D4	483	192	204	90	98.0%	5.7%
D5	978	187	186	113	99.7%	−0.5%
D6	989	219	238	62	100.0%	8.0%
D7	319	162	172	126	99.3%	5.7%
Geometric	20	130	137	140	92.3%	4.4%
Normal	700	279	284	7	97.0%	1.7%
Poisson	20	111	115	172	95.7%	3.1%
Average	628	184	191	104	98.2%	3.6%

efficient and generates better results than Zhu and Wong's for this large N value. Some of results with different N values are shown in Table VI. The averages of DF are always at least 3% larger than Zhu and Wong's. And the averages of DR are all larger than 97%.

6. CONCLUSIONS

We presented a weighted bipartite-matching algorithm for the segmented channel-routing problem. Our approach routes the connections clique by clique, where the connections in each clique are routed using the weighted bipartite-matching method. An approach to successfully judging unroutability is also presented. The average discrimination ratios of the 300 routing cases are 98.8% and 99.4% for the 2-segmentation and 3-segmentation models, respectively. Applying our segmented channel-routing algorithm to the nonunroutable cases, the average of our failure ratios is less than 21% of Zhu and Wong's algorithm.

Table VI. Routability Analysis for the 3-Segmentation with Different N Values

Segmented	N = 400		N = 800		N = 1200		N = 1600	
channel	DR	DF	DR	DF	DR	DF	DR	DF
D1	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%	100.0%	0.0%
D2	100.0%	0.0%	100.0%	0.0%	100.0%	0.4%	100.0%	0.0%
D3	100.0%	3.2%	100.0%	6.3%	99.7%	4.2%	99.3%	3.5%
D4	99.3%	2.5%	99.3%	2.5%	99.3%	1.2%	99.7%	3.3%
D5	100.0%	0.5%	100.0%	0.5%	99.7%	0.0%	100.0%	0.0%
D6	99.0%	21.4%	99.3%	11.7%	99.7%	13.3%	99.0%	17.3%
D7	98.3%	8.5%	98.7%	7.2%	97.7%	8.9%	98.7%	2.7%
Geometric	97.7%	5.9%	99.3%	2.7%	91.3%	5.6%	94.0%	5.1%
Normal	99.3%	3.1%	98.0%	3.8%	97.0%	4.4%	96.7%	2.4%
Poisson	97.3%	7.2%	93.0%	3.2%	95.0%	−0.6%	94.7%	4.2%
Average	99.1%	5.3%	98.8%	3.8%	97.9%	3.7%	98.2%	3.8%

REFERENCES

ACTEL CORP. 1991. *ACT Family FPGA Databook*.

BURMAN, S., KAMALANATHAN, C., AND SHERWANI, N. 1992. New channel segmentation model and associated routing algorithm for high performance FPGAs. In *Proceedings of the 1992 IEEE/ACM International Conference on Computer-Aided Design* (ICCAD '92, Santa Clara, CA, Nov. 8–12), L. Trevillyan, Ed. IEEE Computer Society Press, Los Alamitos, CA, 22–25.

EL GAMAL, A., GREENE, J., AND ROYCHOWDHURY, V. 1991. Segmented channel routing in nearly as efficient as channel routing (and just as hard). In *Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI* (Edimburgh, Scotland), C. H. Séquin, Ed. MIT Press, Cambridge, MA, 192–211.

EL GAMAL, A., GREENE, J., REYNARI, J., ROGOYSKI, E., EL AYAT, K. A., AND MOHSEN, A. 1989. An architecture for electrically configurable gate arrays. *IEEE J. Solid-State Circuits 24*, 2 (Apr.), 394–398.

GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, NY.

GREENE, J., ROYCHOWDHURY, V., KAPTANOGLU, S., AND GAMAL, A. E. 1990. Segmented channel routing. In *Proceedings of the ACM/IEEE Conference on Design Automation* (DAC '90, Orlando, FL, June 24-28), R. C. Smith, Ed. ACM Press, New York, NY, 567–572.

PAPADIMITRIOU, C. H. AND STEIGLITZ, K. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ.

PEDRAM, M., NOBANDEGANI, B. S., AND PREAS, B. T. 1994. Design and analysis of segmented routing channels for row-based FPGA's. *IEEE Trans. Comput.-Aided Des. 13*, 12 (Dec.), 1470–1479.

RHEINBOLDT, W. 1980. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, Inc., New York, NY.

ROY, K. 1993. A bounded search algorithm for segmented channel routing for FPGA's and associated channel architecture issues. *IEEE Trans. Comput.-Aided Des. 12*, 11, 1695–1705.

ROYCHOWDHURY, V. P., GREENE, J., AND EL GAMAL, A. 1993. Segmented channel routing. *IEEE Trans. Comput.-Aided Des. 12*, 1, 79–95.

ZHU, K. AND WONG, D. F. 1992. On channel segmentation design for row-based FPGAs. In *Proceedings of the 1992 IEEE/ACM International Conference on Computer-Aided Design* (ICCAD '92, Santa Clara, CA, Nov. 8–12), L. Trevillyan, Ed. IEEE Computer Society Press, Los Alamitos, CA, 26–29.

Received: July 1997; accepted: March 1999