VLSI DESIGN
1996, Vol. 4, No. 1, pp. 1-10
Reprints available directly from the publisher
Photocopying permitted by license only

A Multi-Terminal Net Router for Field-Programmable Gate Arrays

DINESH BHATIA*
Design Automation Laboratory, Department of Electrical & Computer Engineering and Computer Science, P.O. Box 210030, University of Cincinnati, Cincinnati, OH 45221-0030 (513)-556-2570 (voice) (513)-556-7326 (fax) dinesh.bhatia@UC.EDU

AMIT CHOWDHARY†
Amit Chowdhary Department of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122 amitc@eecs.umich.edu

(Received December 22, 1993, Revised October 18, 1994)

This paper presents a router for routing multi-terminal nets in field-programmable gate arrays (FPGAs). The router does not require pre-assignment of routing channels, a phase that is normally accomplished during global routing. This direct routing approach greatly enhances the probability of routing (routability). The multi-terminal routing greatly reduces the total wire length as it approximates a *Steiner tree*. The total number of segments required to route the circuits is usually less as compared to other routing approaches. The router has generated excellent routing results for some industrial circuits. The memory requirements for this router are very low. The time needed for the routing is *linear* with respect to the size of the circuit.

Key Words: *Routing, Field-Programmable Gate Arrays, VLSI*

1 INTRODUCTION

Recently *Field-programmable gate arrays* (FPGAs) have become very popular for implementing *Application Specific Integrated Circuits* (ASICs). As the technology evolves, the low and medium end ASICs are being implemented using FPGAs. Primarily it is because FPGAs support very rapid prototyping with design realization times of the order of few hours. Due to a restricted architecture, FPGAs have to rely on special purpose CAD tools for design realization. The performance and logic utilization is of primary concern in FPGA based designs. We have addressed, in this paper, the problem of routing nets in an FPGA. As we discuss later, the problem of layout synthesis has not been addressed adequately for FPGAs. We have designed and implemented a performance driven multi-terminal FPGA router for an architecture called *logic cell arrays* (LCAs). The architecture is very popular and has been pioneered by Xilinx [9]. We begin by describing the LCA architecture in the next section.

2 ARCHITECTURE OF AN LCA

The architecture of the logic cell array is depicted in figure 1. An LCA is a two-dimensional array of logic blocks marked as *L*. The LCA also consists of horizontal and vertical routing channels. Each channel consists of sections, that span the length of one logic block. Within the channel sections are present the wiring segments. A channel section (and entire channel) will have *W* wiring segments arranged in parallel tracks. The routing switches are present in the *connection boxes* and *switch boxes*.

A connection box or C box consist of switches that connect the logic block pins to the wiring segments. The switch box or S box switches connect one wiring segments to another. The S boxes are present on the

*Partially supported by the University Research Council of the University of Cincinnati and from Solid State Electronics Directorate, Wright Lab of the US Air Force under contract no. F33615-91-C-1811.
† Partially supported by MTL Systems, Dayton, OHIO.

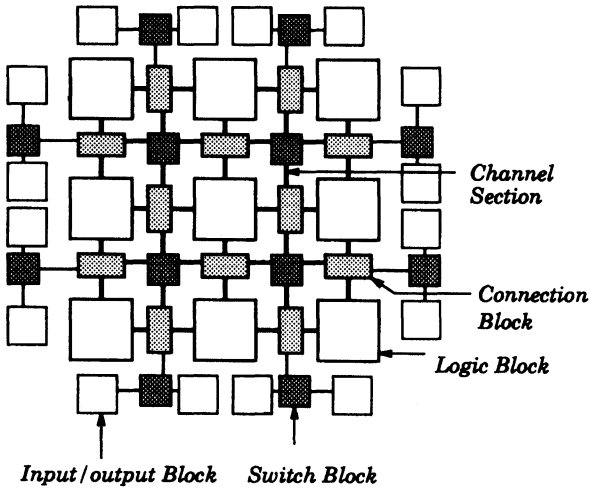

FIGURE 1 Architecture of Logic Cell Array.

intersection of the horizontal and vertical routing channels. The flexibility of a C box, F_c, is defined as the number of segments to which each logic block pin can be connected. The flexibility of an S box, F_s, is defined as the number of segments to which a wiring segment entering a S box can be connected. Figure 2 illustrates a switch box and a connection box and their flexibilities respectively. The flexibilities of C and S boxes are shown in Figure 2. From now on, we will use the terms FPGA and LCA interchangeably.

Brown *et al.* [3] have developed a detailed router for LCAs that routes the two-terminal nets within their assigned global routes. The router, known as the Coarse Graph Expansion (CGE) router, expands the *global route* (coarse graph) of each net to find a detailed route. While routing, the CGE router considers the side-effect of one connection on another. One drawback of the CGE router is that each multi-terminal net is decomposed into a group of two-terminal nets before routing. By doing so the detailed router effectively constructs a *spanning tree* and uses excessive number of wiring segments. In general, a two-terminal net router can result in a wiring where two two-terminal nets belonging to a same multi-terminal net can occupy different segments (tracks) in the same channel. This happens because once the two-terminal nets are formed they are all treated as independent nets. Please refer to Figure 3 for an example where two terminal decomposition of a three terminal net results in two routes that use disjoint set of segments.

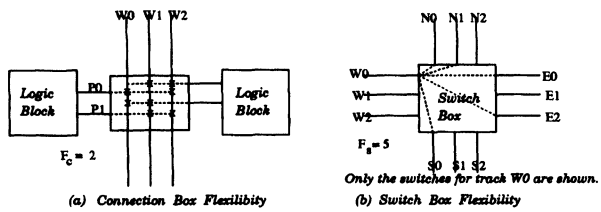

FIGURE 2 The Flexibility of Switch and Connection Boxes.

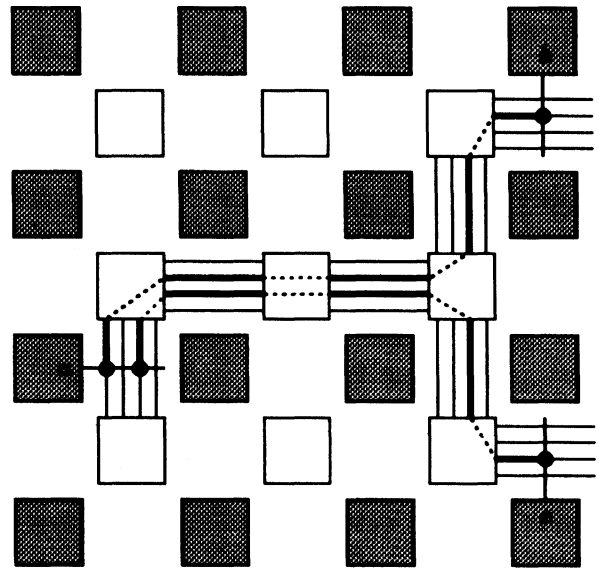

FIGURE 3 An example where two terminal decomposition of a three terminal results in routes that use excessive number of wiring segments. {**a**, **b**, **c**} is a three-terminal net whose two terminal decomposition results into two two-terminal nets {**a**,**b**} and {**a**,**c**} that are routed using disjoint set of segments in common wiring channels.

This type of wiring can result in excessive routing delay especially when the spanning tree is like a *path* (very skewed tree) with an output pin at one end. In this case the maximum delay would depend on the number of input pins in the net. We have observed a few nets in the industrial circuits to have as many as 20 to 30 input pins.

Our router addresses the problem of excessive delays by placing an upper bound on the length of the route from the output pin of a multi-terminal net to any input pin on the same net. This bound is the minimum possible and is obtained from the placement of logic modules. It is easy to see that this bound is the distance between the output pin and the input pin belonging to the same net and farthest from the output pin. We bypass global routing and directly perform the detailed routing after placement. The two stage routing, *i.e.*, global routing followed by the detailed routing is not very efficient. In fact, even in the presence of pre-defined routing architecture, the detailed routing problem in the presence of assigned global routes is NP-complete [8]. The results of Wu and Marek-Sadowska [8] are due to a graph based analysis of the routing architecture of logic cell arrays.

We route multi-terminal nets directly after module placement. By doing so, we not only bypass the global routing but also enhance the probability of routing. As results will show our router is capable of routing large industrial circuits with significantly smaller number of wiring segments. The channel width requirement is almost the same as that for CGE router by Brown *et al.* [2]. In most cases we have observed that 95–99% routing with channel width much less than that required by the

CGE router is easily obtained by our router. Very few nets that are difficult to route in the later stages of routing increase the channel width locally. A manual routing of remaining nets would give significantly improved results.

The multi-terminal router searches the entire space around the pin within the associated bounding box. Also, instead of decomposing the multi-terminal net into two-terminal nets, one input pin of the multi-terminal net is routed to the already routed portion of the net. Thus, the probability of the pin getting routed increases as the routed portion of the net becomes large. Actually, the routing becomes difficult only in the later stages when the majority of the routing resources have been used up, but the increase in the routed portion of the nets more or less compensates for the difficulty in routing the subsequent input pins. The final routing of any one net is an approximation of a *Steiner tree*[1]. It usually requires less number of segments than the route obtained after decomposing multi-terminal nets into two-terminal nets. The performance of the router depends on the topology of the connection box and the switch box (*i.e.* the pattern of the switches in these boxes). Thus, a switch box or a connection box with an efficient topology might result in a smaller channel width.

3 MAIN FEATURES OF THE MULTI-TERMINAL ROUTER FOR FPGAS

The input to the router is the netlist obtained after the placement of modules on an FPGA. Each multi-terminal net n_k is a set of one *output* pin o_k and one or more *input* pins. Thus $n_k = \{o_k, i_k^1, i_k^2, \ldots, i_k^{p_k}\}$, where p_k is the total number of input pins associated with n_k. We distinguish output and input pins from each other. A *net* will refer to a *multi-terminal* net, unless specified otherwise. The aim for routing the nets is to achieve 100% routing with the channel width and the routing delay as low as possible. Currently we bound the routing delay for each net n_k by $\max\{l(o_k, i_k^j)\}$, $1 \leq j \leq p_k$, where $l(a, b)$ is the Manhattan distance between a and b. This metric is less than or equal to the half perimeter of the bounding box for net n_k. Also, due to the fact that all segments are of unit length, the wiring delay due to programmable switches is independent of the track assignment. This is consistent with the XILINX XC2000 and the XC3000 family of FPGAs [9]. In the XC4000 family, segments of different lengths are also available [9]. Thus the wiring delay will have to take both Manhattan distance and track assignment into account. It should be noted that the cost function used for our router can readily accept segments of arbitrary lengths. Besides the primary requirement of routability and small wiring delays, we also address the memory requirements and execution time of our router. It will be evident from the results that the memory and the execution time requirements of the router are extremely small.

The approach normally followed for routing is to decompose each multi-terminal net into a group of two-terminal nets, perform the global routing of two-terminal nets, and then route the circuit within the assigned global routes using a detailed router. A global route for a two-terminal net is a sequence of channel sections from one terminal to another. The search space for finding a detailed route is restricted to the assigned global route. The problem of finding detailed routes within the assigned global routes for FPGAs is NP-complete [8]. We have taken a different approach. In order to make full use of the available routing resources, nets are routed from the netlist description obtained after the placement of the modules, *i.e.*, the global routing is bypassed. We route one input pin of a multi-terminal net at a time to the already routed portion of that net. In other words, the router decides a *Steiner* point for connecting an input pin to the routed portion of the net. Instead of bounding the search space by assigning global routes, the complete feasible space is searched for the routing of each terminal. This results in very high success probability of routing. In our implementation, the input pins, i_k^j, $1 \leq j \leq p_k$, are first ordered in the non increasing order of their Manhattan distance from the output pin o_k belonging to the same net. Let this order be $\{i_k^{j_1}, i_k^{j_2}, \ldots, i_k^{j_{p_k}}\}$. Following this ordering, $i_k^{j_1}$ is routed within distance $l(o_k, i_k^{j_1})$. Subsequently input pins $i_k^{j_2}$ through $i_k^{j_{p_k}}$ are routed to the routed portions of net n_k. For input pins belonging to a net and lying on the timing critical path(s), the net ordering is slightly different. We first route the input pin belonging to a net and lying on a timing critical path to the output pin belonging to the same net. Subsequently, the remaining pins are routed as determined by the net ordering described above. In doing so we gain the following.

1. Each of the p_k input pins of a net n_k is routed in such a way that the length of the path from the output pin $o_k \in n_k$ to an input pin is at most $\{l(o_k, i_k^j)\}$, $1 \leq j \leq p_k$. This places an upper bound on the length of the path between the output pin of a net and any of its input pins. This results in a final routed circuit with a low routing delay. In case of two-terminal routing, there is no upper bound on the length of the path from an output pin to any input pin. In worst, case the path length can be as much as the cost of *minimum*

[1]In practice the multi-terminal router can save at most 33% routing resources. This is due to the fact that Steiner trees are at most 33% better than spanning trees over the same set of vertices [5].

spanning tree obtained for a net n_k using Kruskal's method [6]. In addition, the two terminal decomposition itself is time consuming as it takes $O(p^2)$ time for a net with p terminals.

2. The input pins of a net are routed in the decreasing order of their Manhattan distances from the output pin of the net. Typically the longer paths are difficult to route and initially in the presence of almost all routing resources, they get routed easily. Subsequently, as more and more pins belonging to the same net get routed, the routed portion of the net increases and the probability of routing the subsequent input pins also increases. This is because our router, while routing an input pin, searches for the already routed portions of the net. This compensates for the increase in the difficulty of routing in the later stages of routing.

Figure 4 illustrates a net with an output pin numbered 0 and five input pins numbered 1 to 5. The possible final routes obtained by our router and a two-terminal routing algorithm are also illustrated in figure 4. In a two-terminal routing approach, the net is decomposed into 5 two-terminal nets that are routed in the shortest possible distance. In doing so, the length of the path between the output pin 0 and the input pin 5 becomes very large resulting in a considerable routing delay. The final route obtained by our router as shown in figure 4 results in a *Steiner tree* type of structure. Initially, the input pin 5 is routed to the output pin 0 with the shortest possible distance and then the other pins are routed to the already existing route of that net. The final route thus obtained is shorter than that obtained by the two-terminal routing. The length of a path from the output pin 0 to any input pin is less than the distance between the output pin 0 and the input pin 5. In the example illustrated in Figure 4 the total length of wire for two terminal wiring is 16 while it is 14 for the multi-terminal wiring. Typical industrial circuits can have as many as 40-50 pins, thus two-terminal decomposition can cause significant routing delay.

Our router takes in the topology of the switch box and the connection box (*i.e.* the pattern of the switches) as an input. The topology of a switch box specifies all the pairs of segments on different sides of the switch box that can be connected by programmable switches. Similarly, the connection box topology gives the location of the switches inside the connection box.

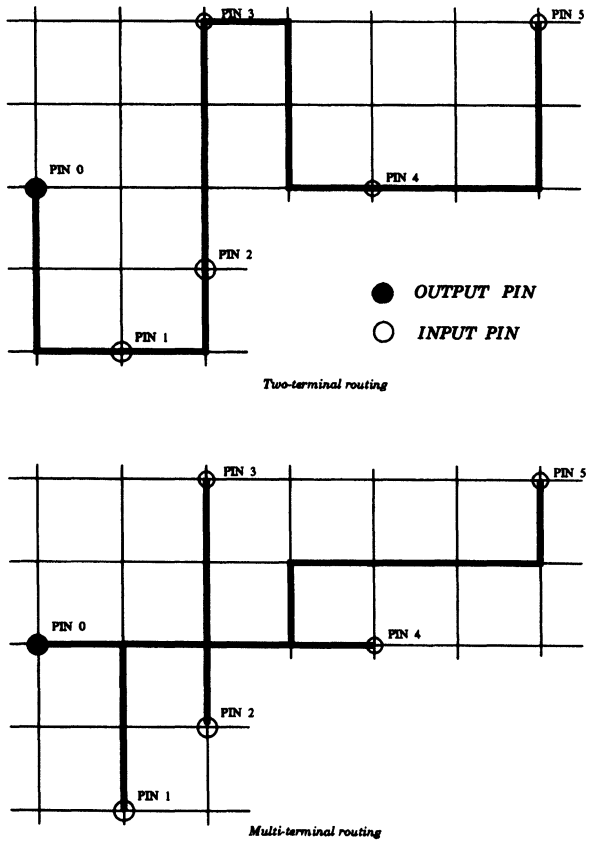

FIGURE 4 Worst case two-terminal routing compared against a multi-terminal routing.

4 MULTI-TERMINAL NET ROUTING

4.1 Terminology

Let N be the total number of nets in a circuit and p_k be the number of input pins in each net n_k, $1 \leq k \leq N$. We denote by $ML(n_k)$, $1 \leq k \leq N$, the distance between o_k and an input pin $i_k^m \in p_k$ such that $l(o_k, i_k^m) = max\{l(o_k, i_k^j)\}$, $1 \leq j \leq p_k$. We denote by $CR(n_k)$, $1 \leq k \leq N$, the partial route of n_k. $CR(n_k)$ includes o_k, a subset of input pins i_k^j, $1 \leq j \leq p_k$, and the wiring segments used to route o_k and the subset of input pins.

A *channel section* is defined as the set of wiring segments between two successive switch-boxes in a horizontal row/vertical column. Two channel sections i and j are said to be adjacent iff they share a common switch box. A *Global-Graph* is a directed acyclic graph $G(V_G, E_G)$ rooted at vertex v_o. A vertex $v_i \in V_G$ represents a channel section i. There exists an edge $v_i v_j$ between two vertices v_i and v_j iff i and j are adjacent channel sections and if v_i belongs to level l in G then v_j belongs to level $l + 1$ in G. Suppose an input pin $i_k^j \in n_k$is to be routed to $CR(n_k)$. Then G is constructed as follows. v_o represents the channel section o such that i_k^j is brought into o using a connection box. Clearly, the *in-degree* of v_o is equal to zero. V_G is a collection of vertices such that:

1. if there exists a wiring segment in channel section

i that belongs to $CR(n_k)$ then v_i belongs to V_G. The vertex v_i is also called a *leaf vertex* in the *Global-Graph*.

2. if there exists a path from i_k^j through a sequence of channel section, $j_1, j_2, \ldots, j_m$, for some $m \leq ML(n_k)$, such that j_m contains a wiring segment that belongs to $CR(n_k)$ then $v_{j1}, v_{j2}, \ldots, v_{jm-1}$ belongs to V_G. Note that v_{jm} will belong to V_G due to condition (1) above. Also v_{ji} , $1 \leq i \leq m$ belongs to level i in G.

The edges of G are constructed as defined above. The *Global-Graph* for net n_k, $1 \leq k \leq N$, consists of $ML(n_k)$ levels, and each level l consists of vertices that represent channel sections that are at distance $l - 1$ from the root vertex representing the channel section where the input pin to be routed exists. The *Global-Graph* associated with any pin belonging to a net lists all the possibilities of routing (all possible global paths). We search for the minimum cost wiring in a *Detailed-Graph* which is obtained after we expand the *Global-Graph* into a *Detailed-Graph*. The *Detailed-Graph* is described below.

To route an input pin $i_k^j \in n_k$, a *Detailed-Graph* $D(V_D, E_D)$ is obtained by expanding each channel section represented in the *Global-Graph* into individual wiring segments. D is also a directed acyclic graph where each vertex $v_i \in V_D$ represents a wiring segment within a channel section. V_D is constructed as follows. Each vertex $v_j \in V_G$ is replaced by a set $\{V_j\}$ of vertices where each vertex represents a feasible wiring segment in channel section j. By feasible wiring segment we imply an unoccupied wiring segment. For *leaf vertices* in the *Global-Graph* the feasible wiring segment set also includes the occupied wiring segments that belong to the net n_k. Also, if $v_j \in V_G$ belongs to level l in G then the corresponding set of vertices $\{V_j\} \in V_D$ also belongs to level l in D. In addition, if v_j is a *leaf vertex* in *Global-Graph*, then $\{V_j\} \in V_D$ represents a set of *leaf vertices*. E_D is constructed as follows. For each v_i, $v_j \in V_G$, if there exists an edge v_iv_j then there exists a directed edge from members of $\{V_i\}$ to members of $\{V_j\}$ if and only if two members share a programmable switch between them in the switch-box between channel sections i and j. Figure 5 illustrates an example. In Figure 5, the *Global-Graph* and corresponding *Detailed-Graph* are shown. The vertices $v_0 \ldots v_9$ in *Global-Graph* represent the channel sections. The vertices in the *Detailed-Graph* represent the feasible wiring segment within a channel section. Finally, we define *routability* as the ratio in percent of the number of nets completely routed to the total number of nets in a given circuit.

4.2 Pre-processing of Nets

Each net n_k has $p_k \geq 1$ input pins. For each net n_k, $1 \leq k \leq N$, the input pins are arranged in a certain predetermined order. As stated earlier, in our implementation, the input pins, i_k^j, $1 \leq j \leq p$, are first ordered in the non increasing order of their Manhattan distance from the output pin o_k belonging to the same net. Let this order be $\{i_k^{j_1}, i_k^{j_2}, \ldots, i_k^{j_{p_k}}\}$. Following this ordering, $i_k^{j_1}$ is routed within distance $l(o_k, i_k^{j_1})$. Subsequently input pins $i_k^{j_2}$ through $i_k^{j_{p_1}}$ are routed to $CR(n_k)$. These pins, *i.e.* $i_k^{j_2}$ through $i_k^{j_{p_1}}$, are not constrained to their bounding boxes with respect to the output pin o_k. Instead pins $i_k^{j_2}$ through $i_k^{j_{p_1}}$ are routed by expansion of global graph into detailed graph when no more than $ML(n_k)$ levels are permitted in the global graph. The ordering of pins should not be mistaken as two terminal decomposition of a multi-terminal net. Here we are arranging input pins in the descending order of their Manhattan distance from the

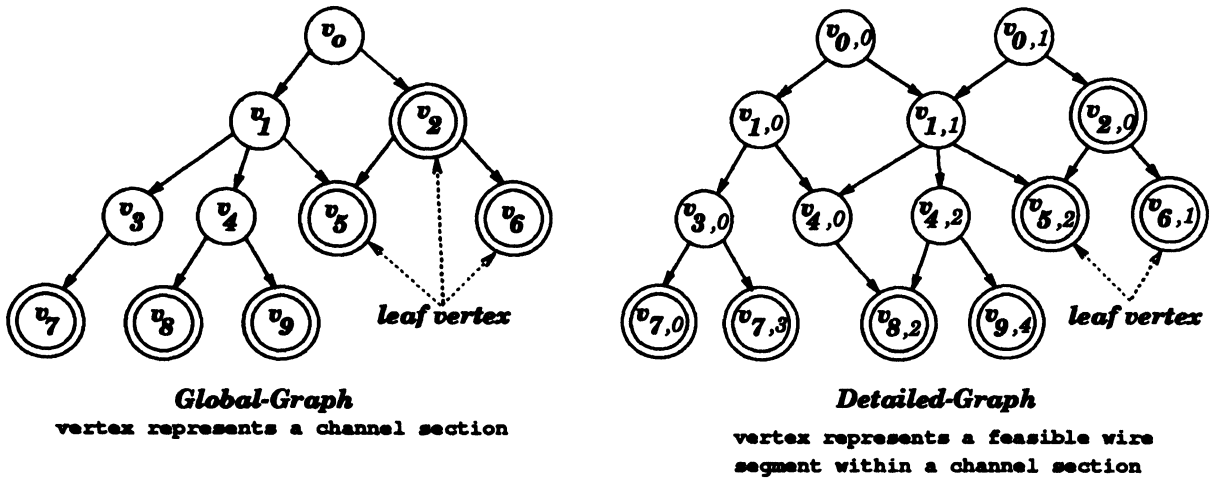

FIGURE 5 The global and detailed graphs.

output pin. Thus for N nets in a circuit, we will have N distinct ordered sequences.

4.3 Routing of nets

We now present the routing algorithm. In each iteration an input pin belonging to a net n_k from each of the ordered sequences is routed to its respective $CR(n_k)$. The algorithm executes $\max\{p_k\}$, $1 \leq k \leq N$ iterations. The routed portion $CR(n_k)$ of each net n_k is small in very early stages of routing. It is likely that few of the input pins might not get routed in the first execution of the algorithm. The algorithm can take multiple passes to improve the routability. Thus if each subsequent pass after first execution takes as input the ordered sequences of unrouted pins, the probability of successful routing would increase due to substantially larger $CR(n_k)$.

Procedure **Route**

Input: A logic cell array, a set of netlists obtained after placement and pin-assignment, a width of routing channel W, and the flexibilities, F_s and F_c, and topologies of switch box and connection box respectively.

Output: A multi-terminal routing if it is feasible with channel width W.

Step 1: For each net n_k, $1 \leq k \leq N$, form the ordered sequence $\{i_k^{j_1}, i_k^{j_2}, \ldots, i_k^{j_{p_k}}\}$ of input pins such that $l(o_k, i_k^{j_{r+1}}), \leq l(o_k, i_k^{j_r})$, $1 \leq r \leq p_k - 1$.

Step 2: For each net n_k such that its ordered sequence of input pins is not empty do Steps 3–7

Step 3: POP i_{n_k} from the top of the ordered sequence of input pins.

Step 4: Construct the *Global-Graph* $G(V_G, E_G)$ with root node representing the channel section in which i_{n_k} belongs.

Step 5: Expand the *Global-Graph* G into a detailed graph $D(V_D, E_D)$.

Step 6: Select the minimum cost leaf node (one with out-degree equal to zero) and trace back a minimum cost path from leaf level to level 1 in D. Add the members of minimum cost path to the $CR(n_k)$.

Step 7: Update cost of all segments and remove the routed input pin $i_{n_k}$from the ordered sequence of input pins for net n_k.

4.4 Cost Function

Each wiring segment in the FPGA has a certain cost assigned to it. After each pin is routed, the cost associated with the wiring segments is updated. The cost assigned to a segment reflects the demand on that segment. The cost of any route is the sum of the costs associated with all the wiring segments that belong to the same route.

We describe the following terms before discussing the cost associated with each wiring segment. Figure 6 illustrates the evaluation of cost function.

1. *Pins-Connectivity* denoted by $PC(i)$ for some pin i is the number of unused wiring segments to which the pin i can be connected to. $PC(i)$ can at most be equal to the connection box flexibility F_c.
2. *Segments-Connectivity* denoted by $SC(i, j)$, for any segment i and $1 \leq j \leq 6$ is the number of segments that a segment i can connect to in one of the six adjacent channel sections. As seen from the Figure 6, a segment in a channel section can connect to segments in six different channel sections through two switch boxes. Initially, the value of this parameter depends solely on the topology of the switch box; *i.e.*, for cases when switch box flexibility, F_s, is a multiple of three, and each incoming segment connects uniformly on all three outgoing sides, then $SC(i, j) = F_s/3$, $1 \leq j \leq 6$. For each segment used in routing, the connectivity of all the segments in the six adjacent channel sections shown in Figure 6 will have to be updated.

The cost of a segment depends on the number of segments in the six adjacent channel sections and the number of pins in the same channel section, that can be connected to the segment, along with their corresponding *Segments-Connectivity* and *Pins-Connectivity*. If a segment in a channel section connects to a large number of segments in the six adjacent channel sections or a large number of pins in the same channel section, then the cost of the segment must be high as it is in a great demand. In another case it is likely that a segment i in a channel

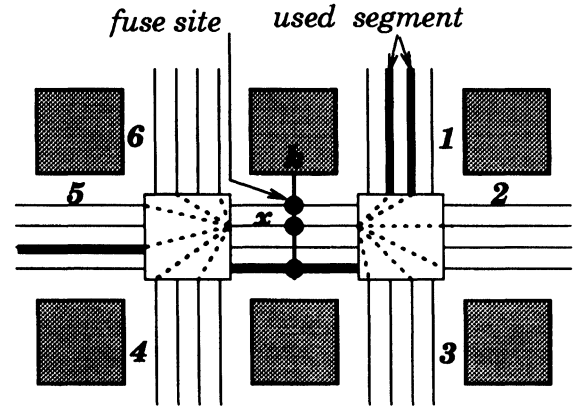

FIGURE 6 An example for evaluating cost function. The cost associated with each segment is dependent upon connectivity of segment in six adjacent channel sections.

section j can connect to a set S of segments in the adjacent channel sections, however the segment belonging to set S can connect to only few segments in channel section j. In such a case, the segment i is in high demand. A similar high cost scenario can exist when a segment i can connect to a set P of pins but for each pin $p \in P$ the $PC(p)$ is very small. In such a case segment i is in high demand. All these facts are taken into account while designing the cost function given below.

$$cost\ [\textbf{segment-no}] = \sum_{j=1}^{6} \sum_{i} \frac{1}{SC(i,j)} + \sum_{k} \frac{\alpha}{PC(k)}$$

Here, the summation over i is done for all the segments in the adjoining channel sections that can be connected to the segment **segment-no**. Similarly, the summation over k is done for all pins in the same segment that can be connected to the segment **segment-no**.

The weight assigned to the cost of a segment due to the pins is taken to be α times the weight of the cost due to the segments in the adjoining channel sections. This is motivated by the fact a pin can be connected in only one channel section, while a segment in a channel section can be connected to segments in six different channel sections. In our experiments, we found that $\alpha = 4$ is a good weighting factor.

4.5 Time Complexity and Memory Requirements of the Router

Lemma 1 *The overall time complexity of the routing is $O(km^2WF_s)$ for an array of size $m \times m$, channel width equal to W, and switch box flexibility of F_s. k is the total number of input pins for the input netlist.*

The time complexity of the routing algorithm is governed by the expansion of the global graph into a detailed graph. For an FPGA of size $m \times m$ and the channel width equal to W, the global graph has $O(m)$ levels and each level has at most m channels. Each channel has W segments, where each segment can connect to at most F_s segments in the next level. Thus, the expansion of global graph into a detailed graph takes $O(m^2 \cdot W \cdot F_s)$ time. We expand a small portion of worst case global graph into a detailed graph for each of the k input pins belonging to the netlist. Thus the overall worst case time complexity is $O(km^2WF_s)$.

Lemma 2 *The worst case space complexity for the router is $O(m^2W)$.*

The memory requirements for our router depends on,

- storing the detailed graph for an input pin, and
- storing the assignment for the routed nets.

The detailed graph has $O(m)$ levels and each level has $O(m)$ channels with each channel having W segments. Thus, the worst case memory required for storing a detailed graph is $O(m^2 \cdot W)$. For storing the routes of all the nets, the memory required is $O(m^2 \cdot W)$. This is true because each segment can be assigned to at most one net and total number of segments is $O(m^2 \cdot W)$. The worst case memory requirements are independent of the number of nets.

5 IMPLEMENTATION DETAILS AND RESULTS

Our router has been used to route several industrial circuits on FPGAs. The FPGA architecture that we used for experiments assumes 4–input lookup table type logic blocks. Each logic block has a D–flipflop. Thus, each logic block has 7 pins: pins 0–3 are the input pins, 4 is the clock pin, 5 is the tri-state pin and 6 is the output pin. Each pin appears on only one side of the logic block. Table 1 shows the number of multi-terminal nets, the number of input pins used and the number of logic blocks used in some of these circuits. These circuits are from different sources: Bell Northern Research (BNR), Zymos and two different designers at University of Toronto (UTD1 & UTD2). In the subsequent subsections, the routing results are presented and compared.

5.1 Channel Density after Global Routing

Before discussing the results obtained after detailed routing for the circuits stated in Table 1, we want to demonstrate the capability of our router for estimating the wiring requirements. Our router can be used as a *global router* for FPGAs. This is done by keeping the switch box flexibility F_s and the connection box flexibility F_c as the maximum possible, *i.e.*, $F_s = 3W$ and $F_c = W$. By doing so, the maximum channel density over all the channels for a circuit or simply the *channel density* will be the lower bound on the number of segments required to complete the detailed routing of a circuit. The

TABLE 1
Characteristics of Experimental Circuits

Circuit	No. of multi-terminal nets	No. of input pins used	No. of logic blocks used	Source	Type
BUSC	151	392	109	UTD1	Bus Cntl
DMA	213	771	224	UTD2	DMA Cntl
BNRE	352	1257	362	BNR	Logic/Data
DFSM	420	1422	401	UTD1	State Mach.
Z03	608	2135	586	Zymos	8-bit Mult

TABLE 2
Channel Density After Global Routing

Circuit	Density from Modified LocusRoute router	Density from Our Router
BUSC	9	8
DMA	10	10
BNRE	11	12
DFSM	10	10
Z03	11	12

channel density for a circuit also reflects the quality of the router used. The channel density for the industrial circuits using our router are shown in Table 2. This channel density is compared with the density obtained after global routing by a modified version of the LocusRoute global router [7]. LocusRoute global router performs global routing for standard cell designs. It has been modified to suit to FPGA routing architecture.

5.2 Channel Width after Detailed Routing

The industrial circuits are routed using our router with the switchbox flexibility, F_s, equal to 6 and the connection box flexiblity, F_c, equal to $0.6W$. The effect of routability for varying switch box and connection box flexiblity has been studied and reported in [1] [4]. The channel width W obtained after routing the circuits using the sequential router is compared with the CGE router [3] in Table 3.

It should be noted from Figures 7 and 8 that the router is capable of routing almost all nets with very small channel widths. In fact, in most cases the routability of as high as 98% is obtained for substantially smaller channel widths. If the remaining nets, as illustrated in Figure 8, are manually routed then we believe that our router can be used for very tight routing. Manual interaction is possible and in most cases we have observed that the routing resources are available that can be used for manual routing.

The channel width obtained after routing using our router depends on the topology of the switch box and the connection box. The topology of these boxes specifies the pairs of segments in the switch box or pin-segment pairs in the connection box that can be connected by a switch. An efficient topology will surely lead to a lower

TABLE 3
Channel Width W Required ($F_s = 6$; $F_c = 0.6W$)

Circuit	W for CGE Router	W for Our Router
BUSC	10	8
DMA	10	11
BNRE	12	12
DFSM	10	11
Z03	13	13

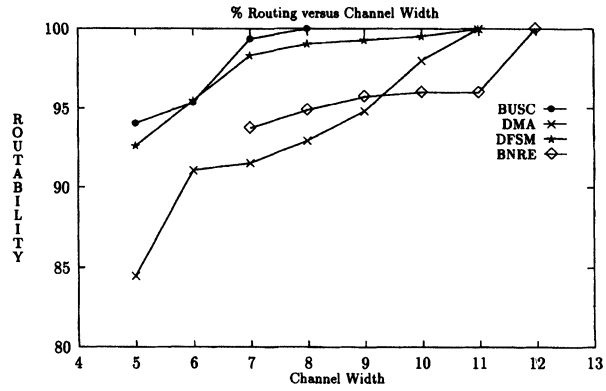

FIGURE 7 The effect of channel width on routability.

channel width. We have observed the effect of switch box topology on the quality of routing. For example, initially the channel width required for routing the DMA circuit was equal to 12. After trying a few different switch box and connection box topologies, the channel width was reduced to 11 which is the minimum possible as obtained by the global routing.

Let W_{CGE} be the channel width required for 100% routing using CGE router. Let R(W) be the ratio of number of nets routed for channel with W to the total number of nets in a circuit using our router. Table 4 shows R(W_{CGE}) for various circuits. About 97–100% of the nets get routed by our router, if the channel width is the minimum required by the CGE router for 100% routing.

5.3 Wiring Segment Utilization

Table 5 shows the total number of wiring segments used for routing the circuits. For $F_s = 6$ and $F_c = 0.6W$, the total wire length in terms of the number of the used segments obtained by our router is always less than that

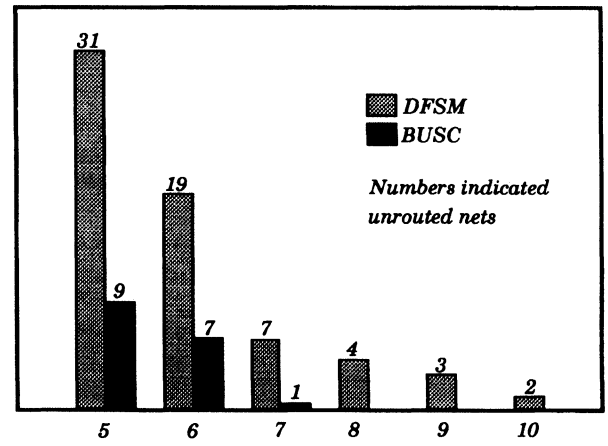

FIGURE 8 Number of unrouted nets for BUSC and DRAM circuits for changing channel widths.

TABLE 4
$R(W_{CGE})$ for Various Circuits.

Circuit	$R(W_{CGE})$
BUSC	1.00, W for our router is 8 and W_{CGE} = 10.
DMA	.98
BNRE	1.00
DFSM	.9952
Z03	1.00

obtained by the CGE router [3]. This is due to the fact that our router results in a *Steiner tree* type of final route for each multi-terminal net. This is usually less than the total length of the net if the net is decomposed into two-terminal nets.

Table 5 also shows the total wire length obtained by using our router as a global router. As mentioned earlier, this is done by keeping the flexibility of the connection box and the switch box as maximum. It should be noted that the two-terminal detailed routes obtained after the global routing always lie within the channels assigned by the global router. Thus, the total wire length estimated after detailed routing is the same as the one obtained after the detailed routing. Thus, total wire length obtained for $F_s = 3$ and $F_c = 0.6W$ using CGE router is also the estimated total wire length obtained after global routing.

5.4 Summary of Results

The results stated in the previous subsections demonstrate the efficiency of our router. For sake of comparison, we have also stated results obtained for the CGE router. A summary of comparison is given below,

- CGE router performs two-terminal routing after global routing. Instead, our router performs multi-terminal net routing in one stage, *i.e.,* no global routing is performed for channel assignment. Input to the router is the placement of a circuit on FPGAs.
- Efficient use of wiring segments during multi-terminal routing. Experimentally, we have found that channel width requirements for the two routers are about the same but the total number of wiring segments used by our router are about 8–10% less than those used by CGE router. Thus for FPGAs, that have restricted number of routing resources, our router performs better in terms of resource utilization.

TABLE 5
Total Number of Wiring Segments Used for 100% Routing

	Our Router		CGE Router
Circuit	Full Flexibility	$F_s = 3$; $F_c = 0.6W$	$F_s = 3$; $F_c = 0.6W$
BUSC	1142	1216	1360
DMA	2507	2606	2862
BNRE	4494	4622	4896
DFSM	4772	4896	5211
Z03	7866	8070	8668

TABLE 6
Execution Time Requirements for Our Router

Circuit	CPU Time (Secs.)
BUSC	8
DMA	32
BNRE	116
DFSM	141
Z03	231

The time taken by our router lies between 8 to 231 CPU seconds as is shown in Table 6. The execution time needed by the router is the CPU time taken on a Sun SPARC-2 work-station. This is almost the same as the time taken for just detailed routing of circuits using the CGE router. The maximum memory requirements for our router never exceeded 500 Kbytes. This is primarily the space required to store all the multi-terminal routes. The reported memory requirements for CGE router lie in between 1.5 to 7.5 megabytes.

6 CONCLUDING REMARKS

We have designed a router for routing multi-terminal nets in field-programmable gate arrays. Our router eliminates the need for the global routing. The multi-terminal routing also bounds the worst case length that a signal has to traverse. Since the detailed routes of the nets have a *Steiner tree* type of configuration, the total number of segments required to route the complete circuit is usually less than any other two-terminal routing approach. Our results are the average case of topologies for switch box and connection box and compare well with the one reported in [3]. It should be noted that the channel width required for 100% routing using our router is for a fully automated design. In practice, we have observed that as many as 98% nets get routed when the channel width is substantially less than that needed for 100% routing.

Acknowledgement

We would like to thank Jonathan Rose and Stephen Brown of the University of Toronto for providing us with the code of the CGE router and the benchmark circuits. We also thank Ms. Akila Subramaniam for helping us in executing the router on the benchmark circuits and Rajasekhar Medicherla for carefully pointing out errors in the manuscript.

References

[1] Dinesh Bhatia, Amit Chowdhary, and Spyros Tragoudas. Mathematical Model for Routability Analysis of FPGAs. In *Proceedings of 4th Great Lakes Symposium on VLSI, IEEE Computer Society Press*, pages 76–79, 1994.

[2] Stephen Brown. *Routing Algorithms and Architectures for Field-Programmable Gate Arrays*. PhD thesis, University of Toronto, 1992.

[3] Stephen Brown, Jonathan Rose, and Zvonko Vranesic. A Detailed Router for Field-Programmable Gate Arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(5):620–627, May 1992.

[4] Stephen D. Brown, Jonathan Rose, and Zvonko Vranesic. A Stochastic Model to Predict the Routability of Field-Programmable Gate Arrays. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 12(12):1827–1838, December 1993.

[5] F. K. Hwang. On Steiner Minimal Trees with Rectilinear Distance. *SIAM Journal on Applied Mathematics, 30:104–114, 1976.*

[6] J. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem. *Proceedings of American Mathematical Society*, 7:48–50, 1956.

[7] J. Rose. Parallel global routing for standard cells. *IEEE Transactions on Computer-Aided Design*, 9:1085–1095, October 1990.

[8] Yu-Liang Wu and M. Marek-Sadowska. Graph Based Analysis of FPGA Routing. In *Proceedings of European Design Automation Conference, EURODAC-93*, pages 104–109, 1993.

[9] Xilinx Inc., San Jose, California. *The Programmable Gate Array Data Book*, 1994.

Biographies

DINESH BHATIA is an Assistant Professor in the department of Electrical and Computer Engineering and Computer Science at the University of Cincinnati. He also directs the Design Automation Laboratory within the same department. Prior to his current position he was visiting Assistant Professor of Computer Science and Engineering at the Southern Methodist University in Dallas. His research interests include the architecture and CAD for field-programmable gate arrays, interconnection problems in VLSI, physical design of MCMs and large ICs, and graph theory and its application in VLSI design.

AMIT CHOWDHARY received the Bachelor of Technology degree in Electrical Engineering from Indian Institute of Technology, Kanpur, India in 1991, and the M.S. degree in Computer Science and Engineering from the University of Cincinnati, Cincinnati, Ohio in 1993. He is currently working towards the Ph.D. degree in Computer Science and Engineering at the University of Michigan, Ann Arbor, Michigan. His main research interests include logic synthesis and architecture of field programmable gate arrays.