New Graph Arborescence and Steiner Constructions for High-Performance FPGA Routing*

Michael J. Alexander and Gabriel Robins

Department of Computer Science, Thornton Hall,
University of Virginia, Charlottesville, VA 22903-2442
Email: robins@cs.virginia.edu
Phone: (804) 982-2207, FAX: (804) 982-2214

Abstract

The flexibility and reusability of field-programmable gate arrays (FPGAs) enable significant speed and cost savings in the VLSI design/validation/simulation cycle. However, this is achieved at a substantial performance penalty due to signal delays through the programmable interconnect. This motivates a critical-net routing objective which seeks to minimize source-sink signal propagation delays; we formulate this objective as a graph Steiner arborescence (i.e., shortest-path tree with minimum wirelength) problem and propose an effective heuristic which produces routing trees with optimal source-sink pathlengths, and having wirelength on par with the best existing graph Steiner tree heuristics. Our second contribution is a new class of greedy Steiner tree constructions in weighted graphs, based on an iterated application of an arbitrary given graph Steiner heuristic; this construction significantly outperforms the best known graph Steiner tree heuristics of Kou, Markowsky and Berman [24], and of Zelikovsky [38]. We incorporated our algorithms into an actual router, enabling the complete routing of several industrial designs while requiring a reduced maximum channel width. All of our methods are directly applicable to other graph-based routing regimes, such as building-block design, routing in the presence of obstacles, etc., as well as in non-CAD areas such as multicasting in communication networks.

1 Introduction

Field-Programmable Gate Arrays (FPGAs) are flexible and reusable high density circuits that can be easily (re)configured by the designer, enabling the VLSI design/validation/simulation cycle to be performed more quickly and cheaply [37]. Unfortunately, the flexibility provided by FPGAs is achieved at a substantial performance penalty due to signal delay through the programmable routing resources, and this is currently a primary concern to both FPGA designers and users [34]. In order to increase FPGA performance, partitioning and technology mapping have been

*Corresponding author is Professor Gabriel Robins, Department of Computer Science, Thornton Hall, University of Virginia, Charlottesville, VA 22903-2442, Email: robins@cs.virginia.edu, phone: (804) 982-2207, FAX: (804) 982-2214.

extensively studied by e.g. [9, 18, 19, 23, 32], where a typical goal is to minimize the maximum circuit depth. On the other hand, less attention has been focused on the actual routing, which is surprising since it was observed that FPGA performance is limited by routing delays, rather than by combinational logic delays [3].

The use of a specific routing resource in routing a particular net precludes its use in routing other nets. In order to maintain the routing feasibility of the overall design, it is therefore prudent to minimize the number of routing resources used by each net. However, for routing *critical nets*, the primary objective is not wirelength optimization, but rather the minimization of source-to-sink delays. (with the secondary criteria being wirelength minimization).

Our first contribution is a method for critical-net routing based on a generalization of rectilinear Steiner arborescences (RSAs) [31] to arbitrary weighted graphs. An arborescence is a shortest-paths tree with minimum wirelength; this minimizes signal delay in a natural way, since shortening pathlengths in the routing reduces the signal travel distance, while the minimum wirelength requirement implies a low overall routing tree capacitance, which again tends to reduce delay. Thus, we generalize Dijkstra's shortest-paths approach [14]: given an arbitrary weighted routing graph, our algorithm produces a spanning tree where all source-sink paths are the shortest possible, but where total tree cost is heuristically optimized as well. Experimental results indicate that the wirelength used is competitive with the best known Steiner tree heuristics.

Our second contribution is a new general class of graph-based greedy algorithms for non-critical-net routing, based on an iterative application of an arbitrary given graph Steiner heuristic. Our Graph Steiner construction significantly outperforms the best-known general Graph Steiner tree heuristics, i.e., those of Kou, Markowsky and Berman [24], and of Zelikovsky [38]. Moreover, the performance bound of our new method is the same as that of Zelikovsky's heuristic, namely $\leq \frac{11}{6}$ times optimal. The recent incorporation of our algorithms into an actual FPGA router [1] enabled the successful routing of several large industry benchmark circuits, using smaller channel widths than was required by other tools. All of our methods are directly applicable to other graph-based routing regimes, such as building-block design, routing in the presence of obstacles, etc., as well as in non-CAD areas such as multicasting in communication networks [25, 29].

The remainder of the paper is organized as follows: Section 2 describes a typical FPGA architecture, reviews previous FPGA routing work, and states the FPGA routing problem for both critical and non-critical nets. Section 3 presents the graph-based Steiner arborescence algorithm for routing critical nets. Section 4 describes the graph Steiner routing method for non-critical-net routing. Section 5 outlines the experimental results, and we conclude in Section 6.

2 FPGA Routing

An FPGA architecture consists of a *symmetrical array* of user configurable logic "blocks", and a set of programmable interconnection resources used for routing [7] [35] (See Figure 1). Each of the logic blocks implements a portion of the design logic, and the routing resources are used to interconnect the logic blocks. Our work focuses on the routing phase of FPGA design; thus, we assume that partitioning, technology mapping, and placement have already been performed.

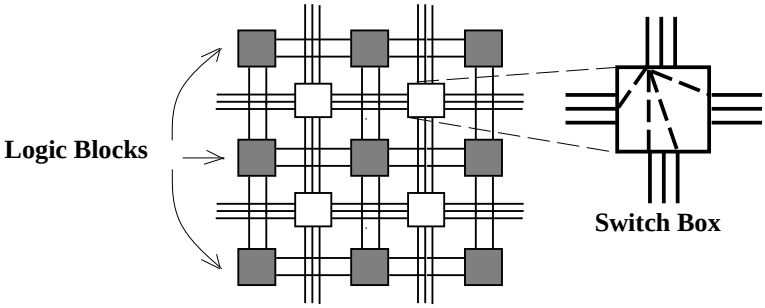

Figure 1: A symmetrical-array FPGA showing some of the logic blocks and programmable interconnection resources.

Previous work on FPGA routing has primarily concentrated on producing feasible solutions that use the fewest routing resources. For example, the SEGA [27] detailed routing algorithm and its predecessor, CGE [6, 7], route nets based on demand and assign critical nets a higher routing priority. Other research has adopted a more abstract model of FPGA routing connections [28], or explored modified architectures [33] in order to reduce the number of programmable switches required. More recently, [1] developed a routing framework where mutually competing objectives (such as congestion, wirelength, and jog minimization) may be simultaneously optimized. Unfortunately, none of these works *directly* minimizes the source-sink signal propagation delays. While these approaches implicitly equate delay minimization with wirelength optimization [15, 21, 22, 30], it recently became increasingly apparent that these two goals are not synonymous [4, 5, 10, 11].

The bounded-radius bounded-cost (BRBC) method of Cong et al. [11] and the AHHK method of Alpert et al. [2] both achieve wirelength-radius tradeoffs in weighted graphs, but can not directly produce a shortest paths tree with minimum wirelength. Rather, with the tradeoff parameter tuned completely towards pathlength minimization, the methods of [11] and [2] both produce the same shortest-paths tree as would Dikstra's algorithm. The recent A-Tree algorithm of Cong et al. [12] for rectilinear arborescence Steiner trees depends heavily on the Manhattan norm, and is therefore not suitable for graph-based regimes.

Before we can apply graph-based techniques to FPGA routing, we must first model the FPGA as a graph, where the overall graph topology mirrors the complete FPGA architecture; paths in this graph correspond to feasible routes on the FPGA, and conversely. Let $G = (V, E)$ denote such a graph, where each graph edge $e_{ij} \in E$ has a weight w_{ij}, which typically corresponds to the wirelength of the associated FPGA routing wire segment (weights may also reflect congestion, jog penalties, etc.). A *net* $N = \{n_0, n_1, ..., n_k\} \subseteq V$ is a set of pins that are to be electrically connected, where n_0 is the signal source and the remaining pins are sinks. A routing solution for a net is a tree $T \subseteq G$ which spans N, and the *cost* of a tree T, denoted $cost(T)$, is the sum of the weights of its edges.

Recall that the high-performance requirement of critical nets dictates a shortest source-sink paths objective, with wirelength minimization being a secondary optimization criteria. For a weighted graph $G = (V, E)$ and two nodes $u, v \in V$, let $minpath_G(u, v)$ denote the *cost* of a shortest path between u and v in G. With this in mind, we formulate the graph Steiner arborescence problem as follows:

The Graph Steiner Arborescence (GSA) Problem: Given a weighted graph $G = (V, E)$, and a net $N \subseteq V$ to be routed in G, construct a *least-cost* spanning tree $T = (V', E')$ with $N \subseteq V' \subseteq V$ and $E' \subset E$ such that $minpath_T(n_0, n_i) = minpath_G(n_0, n_i)$ for all $n_i \in N$.

The complexity of the GSA problem is currently open: no polynomial-time algorithm has been found, and nor is the GSA problem known to be NP-complete [20]. In Section 3 we address the GSA problem using a graph-based generalization of rectilinear Steiner arborescence heuristic of Rao et al. [31].

For non-critical nets, the wirelength minimization objective is crucial in maintaining the feasibility of routing all nets of the design; here we abandon the shortest-paths requirement and state the wirelength minimization problem as follows:

The Graph Minimum Steiner Tree (GMST) Problem: Given a weighted graph $G = (V, E)$, and a net $N \subseteq V$, find a spanning tree $T = (V', E')$ with $N \subseteq V' \subseteq V$ and $E' \subset E$ such that $cost(T)$ is minimum.

Any node in $V - N$ may be used as a potential Steiner point in order to optimize the overall wirelength. The GMST problem is known to be NP-complete [20]; it arises in various applications, e.g., routing in the presence of obstacles [17], FPGA routing [1], as well as in building-block design [8, 16, 26]. In Section 4 we address the GMST problem using an effective greedy strategy.

Figure 2 gives an example of three different routing solutions for a four-pin net (the source is the dark block, while the lightly-shaded blocks are the sinks). Figure 2(a) depicts the solution produced by the KMB graph Steiner heuristic of [24]; Figure 2(b) depicts the optimal Steiner tree solution (which is also the solution produced by our IGMST algorithm described below); and

Figure 2(c) depicts the optimal Steiner arborescence solution (which is also the solution produced by our GSA algorithm described below). Note that KMB uses more wirelength than either of the solutions produced by our two heuristics (IGMST and GSA both produce a wirelength savings of 10% in this example); moreover, the maximum pathlength improvements of IGMST and GSA over KMB in this example are 23% and 45%, respectively.

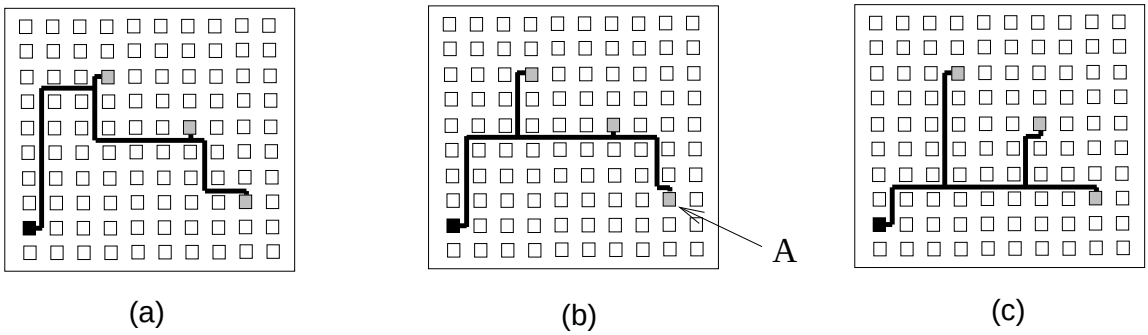

Figure 2: An example of three different routing solutions for a four-pin net (the source is the dark block, while the lightly-shaded blocks are the sinks): (a) the solution produced by the KMB graph Steiner heuristic of [24]; (b) the optimal Steiner tree solution (which is also the solution produced by our IGMST algorithm described below); (c) depicts the optimal Steiner arborescence solution (which is also the solution produced by our GSA algorithm described below). The wirelength saving of IGMST and GSA over KMB in this example is 10%, while the maximum pathlength improvements of IGMST and GSA over KMB are 23% and 45%, respectively.

3 A New Graph Steiner Arborescence Heuristic

For pointsets in the Manhattan plane, a particularly effective heuristic was recently proposed, namely the rectilinear Steiner arborescence (RSA) construction of Rao et al. [31]. The RSA method enjoys a good performance ratio of ≤ 2 times optimal [31], as well as excellent empirical performance. However, the RSA method is strongly dependent on the underlying geometry of the Manhattan metric, and in order to apply it to FPGA routing it must first be generalized to arbitrary weighted graphs.[1]

Before we extend the RSA heuristic to graphs, we first review how it operates in the Manhattan plane (we assume that the source is located at the origin). A point p with coordinates (x_1, y_1) is said to *dominate* point s with coordinates (x_2, y_2) if $x_1 \geq x_2$ and $y_1 \geq y_2$, as shown in Figure

[1]Routing multiple nets on an FPGA causes congestion and blockages among the FPGA routing resources, and these obstructions must be sidestepped by the routes of subsequent nets; thus, the underlying FPGA routing graph structure cannot be modeled by the simple Manhattan plane geometry. This is also why the A-Tree method of Cong et al. [12] is not applicable to arbitrary weighted graphs.

3(a). Define $min(p, q)$ to be the farthest point from the source that is dominated by both p and q (Figure 3(b)). The RSA construction iteratively replaces a pair of points $\{p, q\}$ with the single point $min(p, q)$, where $\{p, q\}$ are chosen as to maximize the distance from the source to $min(p, q)$. The algorithm terminates when only the origin remains, and the final Steiner arborescence solution is formed by connecting each produced $min(p, q)$ to both p and q.

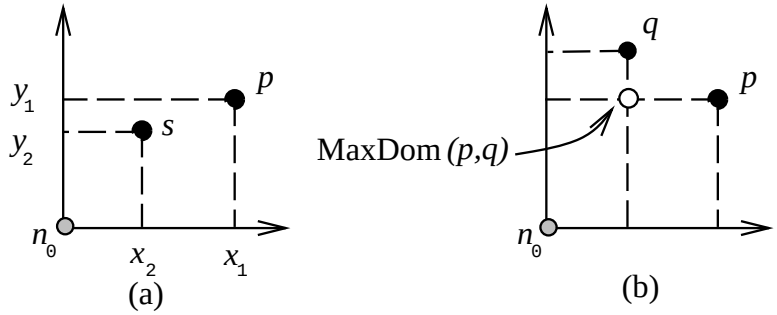

Figure 3: Example of rectilinear dominance: (a) p dominates s, and (b) the point $min(p, q)$.

Generalizing the RSA heuristic to graphs entails extending the concept of dominance to arbitrary weighted graphs. Given a weighted graph $G = (V, E)$, and nodes $\{n_0, p, s\} \subseteq V$, we say that p *dominates* s if:

$$minpath_G(n_0, p) \;=\; minpath_G(n_0, s) \;+\; minpath_G(s, p)$$

This is illustrated in Figure 4(a), where intuitively, p dominates s if a shortest path from the source n_0 to p can pass through s. We define $MaxDom(p, q)$ as a node in V dominated by both p and q which maximizes $minpath_G(n_0, MaxDom(p, q))$ (see Figure 4(b)). $MaxDom(p, q)$ is analogous to $min(p, q)$ in the RSA heuristic described above.

The above definitions enable the following GSA heuristic: starting with the set of nodes that initially contains all sinks, we find a pair of nodes p and q such that $m = MaxDom(p, q)$ is farthest away from the origin among all such pairs; then we replace p and q by m and iterate until only the source remains (see Figure 5). The GSA heuristic can be implemented within time $O(|N| \cdot |E| + |V| \cdot |N|^2 \cdot \log |V|)$. The empirical results in Section 5 indicate that the GSA method is highly effective in producing shortest-paths trees with low wirelength (i.e. wirelength on par with the best existing graph Steiner heuristics).

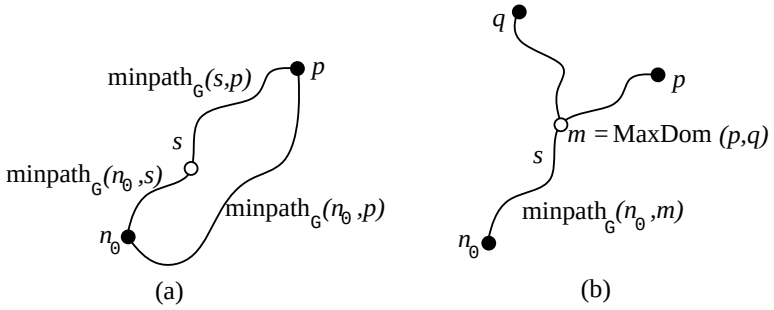

Figure 4: Example of generalized dominance: (a) p dominates s when $minpath_G(n_0, p) = minpath_G(n_0, s) + minpath_G(s, p)$; (b) shows $MaxDom(p, q)$.

Graph Steiner Arborescence (GSA) algorithm
Input: Weighted graph $G = (V, E)$ and net $N \subseteq V$ **Output:** A low-cost shortest-paths tree spanning N

$M = N$
While $N \neq \{n_0\}$ **Do**
  Find a pair $\{p, q\} \subseteq N$ such that $m = MaxDom(p, q)$
  has maximum $minpath(n_0, m)$ over all $\{p, q\} \subseteq N$
  $N = \{N - \{p, q\}\} \cup \{m\}$
  $M = M \cup \{m\}$
Output the tree formed by connecting each node $p \in M$
  (using a path in G) to the nearest node in M that p dominates

Figure 5: The generalized Graph Steiner Arborescence (GSA) heuristic.

4 A New Graph Steiner Tree Heuristic

Recall that in routing non-critical nets we seek to minimize total wirelength, which motivates the graph Steiner tree (GMST) problem discussed in Section 2. A number of heuristics were proposed over the years for the GMST problem [20], two of which have performance bounds of a constant factor from optimal: (1) the heuristic of Kou, Markowsy and Berman [24] (KMB) with performance bound of $2 \cdot (1 - \frac{1}{L})$ where L is the maximum number of leafs in any optimal solution, and (2) the recent heuristic of Zelikovsky [38] (ZEL) with performance bound of $\frac{11}{6}$ (these two methods are described in detail in the Appendix). However, a constant bound from optimality does not automatically imply a good average *empirical* behavior that is significantly better than the bound itself; for example, it is possible that on most typical inputs, a heuristic without *any* bounds on the solution cost could outperform a provably-good heuristic. Similarly, it is possible for one heuristic to significantly outperform another heuristic in practice, even though they both have the *same* theoretical performance bound.

With this in mind, we propose a new class of greedy *iterated* heuristics for the GMST problem. Recall that an instance of the GMST problem is $< G, N >$, where $G = (V, E)$ is a weighted graph, $N \subseteq V$ is a net, and the objective is to find a minimum-cost tree in G that spans N. For any existing graph Steiner tree heuristic H, let $H(G, N)$ denote the solution that H produces with input $< G, N >$, and let $cost(H(G, N))$ denote the cost of that solution.

Our basic algorithm template accepts as input an instance of the GMST problem and any existing GMST heuristic H. It then repeatedly finds Steiner node candidates that reduce the overall spanning cost with respect to H, and includes them into the growing set of Steiner nodes S. More formally, given a set of Steiner candidate node $S \subseteq V - N$, we define the "cost savings" of S with respect to H as follows:

$$\Delta H(G, N, S) = cost(H(G, N)) - cost(H(G, N \cup S))$$

Starting with an initially empty set of Steiner candidates $S = \emptyset$, our heuristic finds a node $t \in V - N$ which maximizes $\Delta H(G, S \cup \{t\}) > 0$ and repeats this procedure with $S \leftarrow S \cup \{t\}$. The cost for H to span $N \cup S$ will decrease with each added node t, and the construction terminates when there is no $t \in (V - N) - S$ such that $\Delta H(G, S \cup \{t\}) > 0$, with the final solution being $H(G, N \cup S)$. This method, which we call the Iterated Graph Minimum Steiner Tree (IGMST) approach, is formally described in Figure 6. Since any existing graph Steiner tree heuristic H may be used in the discussion above, the IGMST method is actually an entire *class* of greedy iterated constructions, one corresponding to each H.

The performance bound of the IGMST method is clearly no worse than the performance bound of the heuristic H that it uses, since if no improving Steiner nodes can be found, the output of IGMST will be identical to the output of H. For example, we may use the KMB graph Steiner heuristic inside the IGMST template as H, to yield the Iterated KMB (IKMB) construction. Similarly, we apply the ZEL heuristic as H inside the IGMST algorithm to produce the Iterated ZEL (IZEL) method. Both the KMB and the ZEL heuristics are described in the Appendix. Since the IKMB solution can be no worse than the KMB solution, the IKMB algorithm inherits the performance bound of the KMB heuristic, namely ≤ 2 times optimal. Similarly, our IZEL construction inherits the theoretical performance bound $\leq \frac{11}{6}$ times optimal of Zelikovsky's heuristic. We also note that the Iterated 1-Steiner heuristic of Kahng and Robins [22] is a special case of IGMST, where H is an ordinary rectilinear minimum spanning tree algorithm. Experimental results in Section 5 indicate that iterating a heuristic H in this fashion yields significantly improved solutions as compared with the non-iterated version of H.

The time complexity of the IGMST heuristics depends on the particular GMST heuristic H that is used. A naive implementation (which treats H as a "black box" subroutine) will have time complexity $O(|N| \cdot |V| \cdot t(H))$, where $t(H)$ is the time complexity of the given GMST heuristic.

Iterated Graph Minimum Steiner Tree (IGMST) Algorithm.
Input: A weighted graph $G = (V, E)$, a net $N \subseteq V$, and a GMST heuristic H **Output:** A low-cost tree $T' = (V', E')$ spanning N, where $N \subseteq V' \subseteq V$ and $E' \subseteq E$
$S = \emptyset$ **Do Forever** $T = \{t \in V - N \mid \Delta H(G, N, S \cup \{t\}) > 0\}$ **If** $T = \emptyset$ **Then Return** $H(G, N \cup S)$ **Find** $t \in T$ with maximum $\Delta H(G, N, S \cup \{t\})$ $S = S \cup \{t\}$

Figure 6: The Iterated Graph Minimum Steiner Tree algorithm (IGMST) using a generic GMST heuristic H.

In practice, this general time complexity may be substantially reduced by extracting out of H common computations such as computing shortest-paths, thereby avoiding duplication of effort among multiple calls to H. For example, IKMB may be implemented within time $O(|N| \cdot |V|^3)$. Another way of reducing the time complexity follows from the observation that rather than adding Steiner points one at a time, they may be added in "batches" based on a *non-interference* criterion [22]. In practice, the number of such rounds tends to be very small (typically ≤ 3), bringing the practical time complexity of IKMB down to $O(|V|^3)$. IZEL may be implemented within time $O(|N|^4 \cdot |V| + |N|^2 \cdot |V|^3)$, but this time bound may be reduced further using the batching idea.

5 Experimental Results

We have implemented the GSA and IGMST algorithms using C in the SUN Unix environment. The code is available from the authors upon request. We have also implemented the KMB and ZEL heuristics (see the Appendix for a detailed description of these), and used each of these as H inside the inner loop of IGMST, yielding the IKMB and IZEL constructions. For comparison, we have also implemented:

- **DJKA** – This heuristic is an adaptation of Dijkstra's shortest-paths tree algorithm [14] to the GSA problem (Dijkstra's algorith spans all of V, while the GSA problem seeks to span only $N \subseteq V$). It first computes a shortest-paths tree rooted at the source using Dijkstra's algorithm, and then deletes edges from this tree which are not contained in any source-to-sink path; and

- **DOM** – This heuristic is a restricted version of the GSA heuristic in Figure 5, where $Maxdom(p, q)$ is constrained to be only nodes from N, rather than an arbitrary node. Intuitively, an approximate arborescence is constructed by connecting each sink to the closest

node that it dominates. This is a graph *spanning* arborescence (as opposed to a *Steiner* arborescence).

We compared all of these methods (i.e., DJKA, DOM, GSA, KMB, ZEL, IKMB, IZEL) on the same inputs, both in terms of total wirelength as well as maximum source-sink pathlength. The inputs consisted of channel intersection graphs [13] over uniformly distributed random nets in the 10000×10000 Manhattan grid. For each net size, 10000 random nets were generated and routed using the seven algorithms above.

The data in Table 1 represents average values over 10000 cases per each net size. For each net, we normalized the wirelength produced by each heuristic with respect to the wirelength of the KMB algorithm; similarly, the maximum source-sink pathlength of each heuristic was normalized with respect to that of GSA (i.e., the optimal radius). The table numbers represent average percent improvement; a *positive* value represents an *increase* in the maximum wirelength (resp. radius) with respect to KMB (resp. GSA), while a *negative* number represents a decrease (i.e., win).

	Average Wirelength and Radius Statistics (in percent)									
	3-pin nets		**4-pin nets**		**6-pin nets**		**8-pin nets**		**10-pin nets**	
Algorithm	Wire Length (w.r.t. KMB)	Max Radius (w.r.t. DJKA)	Wire Length (w.r.t. KMB)	Max Radius (w.r.t. DJKA)	Wire Length (w.r.t. KMB)	Max Radius (w.r.t. DJKA)	Wire Length (w.r.t. KMB)	Max Radius (w.r.t. DJKA)	Wire Length (w.r.t. KMB)	Max Radius (w.r.t. DJKA)
DJKA	10.75	0.00	17.33	0.00	26.70	0.00	34.62	0.00	41.13	0.00
DOM	6.94	0.00	11.03	0.00	16.33	0.00	20.58	0.00	23.48	0.00
GSA	**-3.29**	**0.00**	**-4.71**	**0.00**	**-5.27**	**0.00**	**-4.91**	**0.00**	**-4.21**	**0.00**
KMB	0.00	7.77	0.00	15.77	0.00	26.67	0.00	37.21	0.00	44.02
ZEL	-3.29	0.00	-4.47	4.30	-5.67	11.44	-6.01	19.10	-6.19	27.02
IKMB	**-3.29**	**0.00**	**-4.71**	**4.06**	**-6.11**	**9.83**	**-6.62**	**16.22**	**-6.87**	**23.33**
IZEL	**-3.29**	**0.00**	**-4.71**	**3.41**	**-6.32**	**9.37**	**-6.93**	**15.42**	**-7.23**	**22.66**

Table 1: The average wirelength and average maximum radius produced by the seven algorithms for various net sizes. The wirelength values are normalized with respect to KMB, while the maximum-radius values are normalized to the (optimal) values produced by GSA. Negative values represent improvement (i.e., savings), while positive values represent disimprovement.

In terms of wirelength, we observe that across all net sizes, DJKA has the worst performance since it uses up to 41% more wirelength than KMB. DOM performs somewhat better, losing up to 23% wirelength to KMB. On the other hand, GSA *outperforms* KMB in term of wirelength by up to 5%. This is a very significant observation, since KMB is designed to minimize wirelength only, yet it consistently loses in terms of wirelength to GSA, a construction which also has optimal radius.

In terms of maximum source-sink pathlength, we consistently observe that KMB substantially lags behind ZEL (by up to 44%); on the other hand, IKMB outperforms ZEL by up to 3%, and IZEL has the best performance among all the heuristics, but only slightly better than that IKMB. Interestingly, the relative performance ordering KMB $<$ ZEL $<$ IKMB $<$ IZEL is *consistent* in terms of *both* wirelength *and* maximum radius, across all net sizes.

For nets of size 4, GSA produces routings with optimal wirelength *and* optimal radius. This is significant because it is known that for typical VLSI designs most nets have four or less pins [17]. Finally, we see that on average across all nets, GSA affords optimal radius (i.e., 10% average radius reduction with respect to IZEL) at the expense of an average of only about 1% wirelength penalty. We conclude from this that GSA affords a very favorable radius-wirelength tradeoff in that it yields high-performance routings at a negligible wirelength penalty. Thus by using GSA to route an FPGA, the routability of the design will not be significantly affected, while its overall performance would be substantially improved.

A common criteria used to evaluate the quality of FPGA routing solutions is the maximum channel width required to successfully route all nets of a design [6]. We have therefore implemented a real FPGA router based on our algorithms, and used it to route several industrial FPGA benchmark circuits (the underlying architecture we modeled was that of the XILIX parts, and the benchmark circuits contained up to 352 nets each). For each of the circuits, we compared the maximum channel width required by our router to that required by CGE [6]. Table 2 shows the maximum channel width required by the CGE router [6], as well as the corresponding value for our IGMST-based router. Note that our IGMST-based router is able to route all of the benchmark circuits using *fewer* routing resources than CGE, indicating that even when routing *multiple* nets on an actual FPGA topology, IGMST uses substantially less wirelength to route all the nets. Figure 7 illustrates the solution produced by IGMST for the smallest of the benchmark circuits.

Percentage of nets with 10 pins or less				Maximum channel width required for a feasible routing of all nets	
Circuit	Total #nets	#nets with 10 pins or less	Percentage of Total	CGE	**IGMST**
BUSC	151	143	94.7	10	**8**
DMA	213	191	89.7	10	**9**
BNRE	352	325	92.3	12	**11**

Table 2: Results for a set of industry FPGA benchmark circuits; note that the majority of nets contain 10 or fewer pins. On the right, we see the maximum channel width required to route all nets for the CGE router of [6], as well as our IGMST heuristic; for all of the circuits IGMST requires a smaller channel width (i.e., *fewer* routing resources).

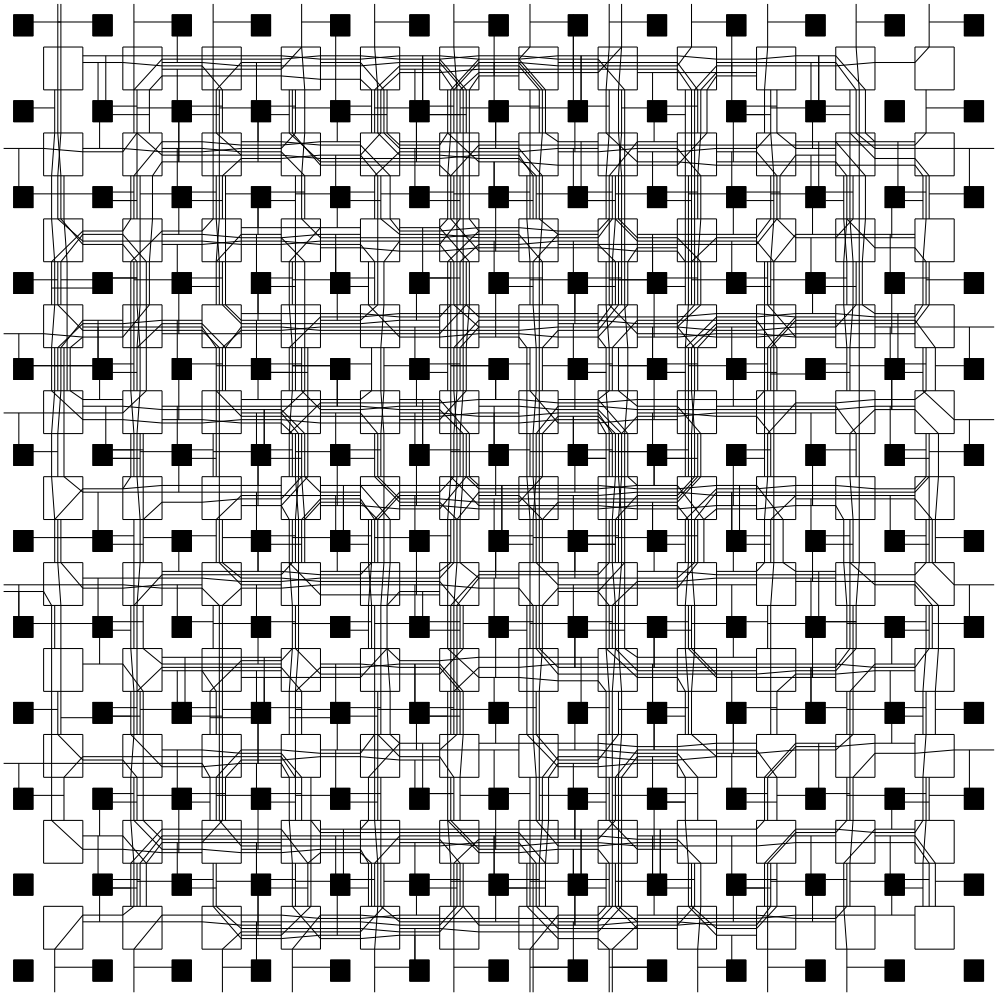

Figure 7: Solution produced by IGMST-based router for BUSC, the smallest of the three benchmark circuits.

6 Conclusion

We have proposed a critical-net FPGA routing algorithm to mitigate the performance penalty incurred by designers when using FPGAs. Our approach entails constructing shortest-paths trees with minimum wirelength, based on a generalization to graphs of a rectilinear Steiner arborescence heuristic. Our method produces routing trees with optimal source-sink pathlengths, while using total wirelength that is competitive with the best existing graph Steiner tree heuristics. For non-critical-net routing where the objective is exclusively wirelength minimization, we offer an effective new class of iterative graph Steiner tree constructions that improve upon the performance of an

arbitrary given graph Steiner heuristic without sacrificing the theoretical performance bound of the heuristic used. All of our methods are directly applicable to other routing regimes, e.g., building-block design, routing in the presence of obstacles, etc., as well as to non-CAD areas such as multicasting in communication networks.

References

[1] M. J. Alexander and G. Robins, *A New Approach to FPGA Routing Based on Multi-Weighted Graphs*, in Proc. ACM/SIGDA International Workshop on Field-Programmable Gate Arrays, Berkeley, CA, February 1994.

[2] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng, *A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Global Routing*, in Proc. IEEE Intl. Symp. Circuits and Systems, Chicago, IL, May 1993, pp. 1869–1872.

[3] N. B. Bhat and D. D. Hill, *Routable Technology Mapping for LUT FPGAs*, in Proc. IEEE Intl. Conf. Computer-Aided Design, 1992, pp. 95–98.

[4] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, *Fidelity and Near-Optimality of Elmore-Based Routing Constructions*, in Proc. IEEE Intl. Conf. Computer Design, Cambridge, MA, October 1993, pp. 81–84.

[5] K. D. Boese, A. B. Kahng, and G. Robins, *High-Performance Routing Trees With Identified Critical Sinks*, in Proc. ACM/IEEE Design Automation Conf., Dallas, June 1993, pp. 182–187.

[6] S. Brown, J. Rose, and Z. G. Vranesic, *A Detailed Router for Field-Programmable Gate Arrays*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 620–628.

[7] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, Boston, MA, 1992.

[8] D. Chen and C. Sechen, *Mickey: A Macro Cell Global Router*, in Proc. European Design Automation Conf., Amsterdam, The Netherlands, February 1991, pp. 248–252.

[9] K. C. Chen, J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, *DAG-Map: Graph-Based FPGA Technology Mapping for Delay Optimization*, IEEE Design & Test of Computers, 9 (1992), pp. 7–20.

[10] J. Cohoon and J. Randall, *Critical Net Routing*, in Proc. IEEE Intl. Conf. Computer Design, Cambridge, MA, October 1991, pp. 174–177.

[11] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, *Provably Good Performance-Driven Global Routing*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 739–752.

[12] J. Cong, K. S. Leung, and D. Zhou, *Performance-Driven Interconnect Design Based on Distributed RC Delay Model*, in Proc. ACM/IEEE Design Automation Conf., Dallas, June 1993, pp. 606–611.

[13] W. M. Dai, T. Asano, and E. S. Kuh, *Routing Region Definition and Ordering Scheme for Building-Block Layout*, IEEE Trans. Computer-Aided Design, 4 (1985), pp. 189–197.

[14] E. W. Dijkstra, *A Note on Two Problems in Connection With Graphs*, Numerische Mathematik, 1 (1959), pp. 269–271.

[15] A. E. Dunlop, V. D. Agrawal, D. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, *Chip Layout Optimization Using Critical Path Weighting*, in Proc. ACM/IEEE Design Automation Conf., 1984, pp. 133–136.

[16] H. Esbensen and P. Mazumder, *A Genetic Algorithm for the Steiner Problem in a Graph*, in Proc. European Design Automation Conf., Paris, France, February 1994, pp. 402–406.

[17] J. L. Ganley and J. P. Cohoon, *Routing a Multi-Terminal Critical Net: Steiner Tree Construction in the Presence of Obstacles*, in Proc. IEEE Intl. Symp. Circuits and Systems (to appear), London, England, May 1994.

[18] T. Gao, K. C. Chen, J. Cong, Y. Ding, and C. L. Liu, *Placement and Placement Driven Technology Mapping for FPGA Synthesis*, in Proc. IEEE Intl. ASIC Conf., Rochester, NY, September 1993, pp. 87–91.

[19] N. Hasan, G. Vijayan, and C. K. Wong, *A Neighborhood Improvement Algorithm for Rectilinear Steiner Trees*, in Proc. IEEE Intl. Symp. Circuits and Systems, New Orleans, LA, 1990.

[20] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*, North-Holland, 1992.

[21] M. A. B. Jackson, E. S. Kuh, and M. Marek-Sadowska, *Timing-Driven Routing for Building Block Layout*, in Proc. IEEE Intl. Symp. Circuits and Systems, 1987, pp. 518–519.

[22] A. B. Kahng and G. Robins, *A New Class of Iterative Steiner Tree Heuristics With Good Performance*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 893–902.

[23] K. Karplus, *Xmap: a Technology Mapper for Table-lookup Field-Programmable Gate Arrays*, in Proc. ACM/IEEE Design Automation Conf., 1991, pp. 240–243.

[24] L. Kou, G. Markowsky, and L. Berman, *A Fast Algorithm for Steiner Trees*, Acta Informatica, 15 (1981), pp. 141–145.

[25] Y. Lan, A. H. Esfahanian, and L. M. Ni, *Distributed Multi-Destination Routing in Hypercube Multiprocessors*, in Proc. 3rd Conf. Hypercube Computers and Concurrent Applications, January 1988, pp. 631–639.

[26] K. W. Lee and C. Sechen, *A Global Router for Sea-of-Gates Circuits*, in Proc. European Design Automation Conf., Amsterdam, The Netherlands, February 1991, pp. 242–247.

[27] G. G. Lemieux and S. D. Brown, *A Detailed Routing Algorithm for Allocating Wire Segments in Field-Programmable Gate Arrays*, in Proc. ACM/SIGDA Physical Design Workshop, Lake Arrowhead, CA, April 1993.

[28] F. D. Lewis and W. C. Pong, *A Negative Reinforcement Method of PGA Routing*, in Proc. ACM/IEEE Design Automation Conf., 1993, pp. 601–605.

[29] X. Lin and L. M. Ni, *Multicast Communication in Multicomputer Networks*, IEEE Trans. Parallel and Distributed Systems, 4 (1993), pp. 1105–1117.

[30] S. Prasitjutrakul and W. J. Kubitz, *A Timing-Driven Global Router for Custom Chip Design*, in Proc. IEEE Intl. Conf. Computer-Aided Design, Santa Clara, CA, November 1990, pp. 48–51.

[31] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor, *The Rectilinear Steiner Arborescence Problem*, Algorithmica, (1992), pp. 277–288.

[32] K. Roy, B. Guan, and C. Sechen, *FPGA MCM Partitioning and Placement*, in Proc. ACM/SIGDA Physical Design Workshop, Lake Arrowhead, CA, April 1993, pp. 211–212.

[33] Y. Sun, T. C. Wang, C. K. Wong, and C. L. Liu, *Routing for Symmetric FPGAs and FPICs*, in Proc. IEEE Intl. Conf. Computer-Aided Design, Santa Clara, CA, November 1993, pp. 486–490.

[34] S. Trimberger, Manager of Advanced Development, Xilinx Inc., private communication, February, 1994.

[35] S. Trimberger, *Field-Programmable Gate Arrays*, IEEE Design & Test of Computers, 9 (1992), pp. 3–5.

[36] Y. F. Wu, P. Widmayer, and C. K. Wong, *A Faster Approximation Algorithm for the Steiner Problem in Graphs*, Acta Informatica, 23 (1986), pp. 223–229.

[37] Xilinx, *The Programmable Gate Array Data Book*, Xilinx, Inc., San Jose, California, 1991.

[38] A. Z. Zelikovsky, *An 11/6 Approximation Algorithm for the Network Steiner Problem*, Algorithmica, 9 (1993), pp. 463–470.

7 Appendix: Two Existing Graph Steiner Heuristics

This appendix summarizes the graph Steiner heuristics of (1) Kou, Markowsky, and Berman (KMB) [24], and (2) Zelikovsky (ZEL) [38], the currently best known Graph Steiner heuristics. KMB has a theoretical performance ratio[2] of $2 \cdot (1 - \frac{1}{L})$, where L is the maximum number of leaves in any optimal (Steiner tree) solution to the input instance. The ZEL heuristic has performance ratio of $\frac{11}{6}$, and thus on worst-case instances ZEL will outperform KMB.[3]

Any graph Steiner heuristic H may be used inside the IGMST template of Section 4 to yield an improved approximation algorithm for the IGMST problem. The theoretical performance bound of the composite construction is guaranteed to be no worse than that of the heuristic H that was used in the iterated construction. Thus, the performance bound of the IKMB algorithm is $2 \cdot (1 - \frac{1}{L})$, where L is the maximum number of leaves in any optimal solution, while the performance bound of IZEL is $\frac{11}{6}$.

7.1 The KMB Heuristic [24]

The graph Steiner tree heuristic of Kou, Markowsky and Berman (KMB) [24] is described as follows (see Figure 8):

- Construct the *distance* graph G' over N as follows: form the complete graph G' over N with the weight of each edge e_{ij} equal to $dist_G(n_i, n_j)$, i.e., the cost of the corresponding shortest path in G between n_i and n_j.
- Compute $MST(G')$, the minimum spanning tree of G', and expand each edge e_{ij} of $MST(G')$ into the corresponding shortest path, denoted $path(n_i, n_j)$, yielding a subgraph G'' that spans N.
- Finally, compute the minimum spanning tree $MST(G'')$, and delete pendant edges from $MST(G'')$ until all leaves are members of N.

The time complexity of the KMB heuristic in Figure 8 is $O(|N| \cdot |V|^2)$; recently, this has been reduced to $O(|E| \cdot \log|V|)$ using an alternative implementation [36].

7.2 The ZEL Heuristic [38]

Define a *triple* to be a set of three nodes in N; we can *contract* a graph around a triple z by setting to zero the edge weights of two of the three edges connecting nodes of the triple (the contracted graph is denoted by $G'[z]$). The ZEL heuristic of Zelikovsky [38] is described as follows:

[2] The *performance ratio* of a heuristic is the worst-case ratio of its solution with respect to the optimal solution; for example, a heuristic having performance ratio of $\frac{3}{2}$ is guaranteed to produce solutions with cost never more than 50% more than the optimal.

[3] Note that in a practical setting, the performance ratios of KMB and ZEL are incomparable, since although KMB can produce solutions with cost arbitrarily close to twice the optimal, for some inputs it can produce solutions that are much better than twice the optimal (i.e., on inputs where L is small). Thus it is not clear a priori from these theoretical bounds that KMB will consistently lose to ZEL on typical nets (although the empirical data in Section 5 indicates that this is indeed the case).

The Kou, Markowsky and Berman (KMB) Algorithm [24]
Input: A graph $G = (V, E)$ with edge weights w_{ij} and a net $N \subseteq V$ **Output:** A low-cost tree $T' = (V', E')$ spanning N (i.e. $N \subseteq V' \subseteq V$ and $E' \subseteq E$)
$G' = (N, N \times N)$, with edge weights $w'_{ij} = dist_G(n_i, n_j)$ **Compute** $T = (N, E'') = \text{MST}(G')$ $G'' = \cup_{e_{ij} \in E''} path_G(n_i, n_j)$ **Compute** $T' = \text{MST}(G'')$ **Delete** pendant from T' until all leaf nodes that are not in N **Output** T'

Figure 8: The KMB heuristic for the GMST problem [24].

- Construct the distance graph G' over N (See the first bullet of Section 7.1).
- For every triple $z \in N$, find $v \in V$ which minimizes $\sum_{s \in z} dist_G(s, v)$; (i.e., the Steiner point which will produce the greatest savings for each triple). Record these values as $v_z = v$ and $dist_z$, respectively.
- Find $z \in N$ which maximizes win, where $0 \leq win = MST(G') - MST(G'[z]) - dist_z$, and contract G' around z to get a new $G' \leftarrow G'[z]$. The Steiner point associated with the contracting triple, v_z, becomes a Steiner point in the solution. Repeat this greedy contraction step while $win \geq 0$.
- Construct a solution using the KMB algorithm where the nodes to be spanned are the original nodes N plus the v_z associated with the $G'[z]$ contractions above.

The time complexity of the ZEL heuristic is $O(|V| \cdot |E| + |N|^4)$.

The Zelikovsky (ZEL) Algorithm [38]
Input: A graph $G = (V, E)$ with edge weights w_{ij} and a net $N \subseteq V$ **Output:** A low-cost tree $T' = (V', E')$ spanning N (i.e. $N \subseteq V' \subseteq V$ and $E' \subseteq E$)

$G' = (N, N \times N)$, with edge weights $w'_{ij} = dist_G(n_i, n_j)$,
$W = \emptyset, Triples = \{z \subset N : \lvert z \rvert = 3\}$
For every $z \in Triples$ **Do**
  Find v which maximizes $\sum_{s \in z} dist_G(s, v)$.
  $v_z = v$ and $dist_z = \sum_{s \in z} dist_G(s, v)$
Repeat Forever
  Find $z \in Triples$ which maximizes $win = MST(G') - MST(G'[z]) - dist_z$
  If $win \leq 0$ **Then Return** $\text{KMB}_G(N \cup W)$
  $G' = G'[z]$
  $W = W \cup v_z$

Figure 9: The ZEL heuristic for the GMST problem [38].