A Detailed Routing Algorithm for Allocating Wire Segments in Field-Programmable Gate Arrays

Guy G. Lemieux and Stephen D. Brown
Department of Electrical and Computer Engineering
University of Toronto, Canada

Abstract

This paper describes a new detailed routing algorithm that has been designed specifically for the types of routing architectures that are found in the most recent generation of Field-Programmable Gate Arrays (FPGAs). The router is intended for FPGAs that fit within the symmetrical category, which means that the architecture consists of rows and columns of logic cells with both vertical and horizontal routing channels. The routing algorithm, called SEGA, is unique in that it focuses on not only the issue of achieving successful routing of 100 percent of the required connections for a circuit, but also addresses the allocation of wire segments to connections in a way that matches the lengths of the segments to the lengths of the connections. The implementation of the SEGA program is designed in a way that supports a wide range of routing architectures, making the algorithm useful as a research vehicle for exploring new architectures for future FPGAs.

SEGA has been used to obtain excellent routing results over a set of several benchmark circuits. The results show that the algorithm can route all of the circuits tested in very close to the theoretical minimum number of routing tracks per channel. In addition, SEGA has been shown to adeptly allocate the wire segments in a segmented FPGA according to the lengths of connections. More specifically, the router does a good job of limiting the number of segments used for long connections and the length of segments assigned to short connections.

1 Introduction

Over the past several years, Field-Programmable Gate Arrays (FPGAs) have become widely accepted as an attractive means of implementing digital circuits in a customized VLSI chip. While a number of classes of FPGAs are commercially available, one of the most important types is the symmetrical FPGA, which consists of rows and columns of logic blocks with horizontal routing channels between the rows and vertical channels separating the columns. This class of FPGA was first introduced by Xilinx in [1], and later chips were described in [2], and [3]. In addition, the symmetrical architecture is found in FPGAs produced by QuickLogic [4] and (very recently) Altera [5].

Symmetrical FPGAs are available with very high logic capacities, approaching the equivalent of 20,000 basic logic gates. With such large devices, the design of the interconnect in the routing channels has a crucial impact on both the percentage of the chip's logic capacity that can be effectively utilized and the speed-performance of circuits implemented in the FPGA. In early symmetrical FPGAs [1] [2], the interconnect comprised mostly short wire segments that spanned the length or width of a single logic block. Longer wire segments could be formed by interconnecting the short segments via programmable routing switches. While this architecture allows for efficient utilization of the wire segments in terms of area (since short connections never waste area by using long wire segments), requiring that long connections pass through several routing switches in series impairs speed-performance. This follows because FPGA routing switches are user-programmable and hence have significant series resistance and parasitic capacitance. To

address these issues, more recent architectures contain a mixture of both short and long wire segments — short segments minimize wasted area for short connections, and long segments minimize propagation delay for longer connections.

It is clear that implementing any non-trivial circuit in a complex FPGA requires sophisticated Computer-Aided Design (CAD) tools. Although the number of software stages involved in mapping a circuit into an FPGA varies somewhat from one vendor to the next, the following steps [6] are typically involved: initial design entry, logic optimization, technology mapping, placement, and routing. This paper focuses on routing, the final stage in the CAD process, and assumes that the preceding steps have already been carried out using some reasonable algorithms. The paper describes a new detailed routing algorithm, called *SEGA* (for SEGment Allocator), that has been designed specifically for the types of routing architectures found in the most recent generation of symmetrical FPGAs. SEGA is designed to balance the three issues that are most important with segmented routing channels, namely that 100 percent of the required connections must be routed, that long wire segments should not be wasted on short connections, and that long connections should be routed with appropriately long wire segments. SEGA is unique in that it addresses all of these issues, and it is parameterized to be usable over a wide range of FPGA routing architectures that fit the general model of the symmetrical FPGA.

1.1 Motivation for Developing a New Algorithm

Since numerous routing algorithms for VLSI chips have been created over the years, it is prudent to explain why a new algorithm is desirable for symmetrical FPGAs with segmented routing channels. To begin with, detailed routing in FPGAs with any style of routing architecture can be more difficult than classical detailed routing [7] [8] because the segments available for routing are already in place and joins between segments are possible only where routing switches exist. To illustrate the difficulties involved, consider the example described below.

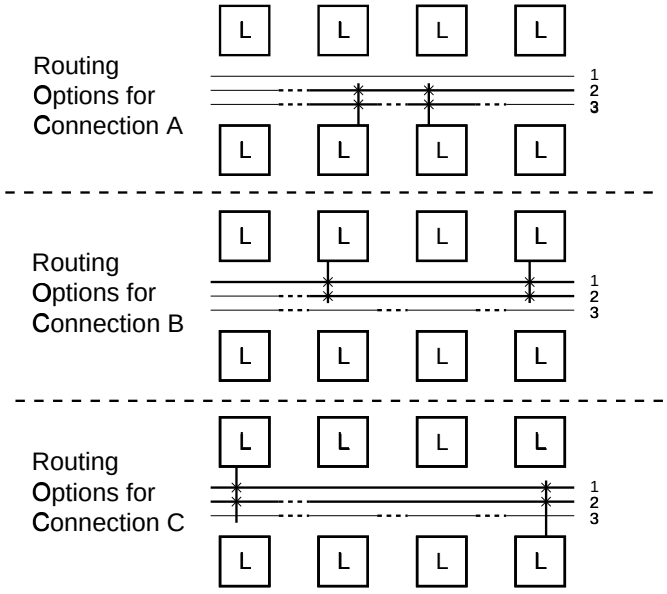

Figure 1 - *An Example of an FPGA Routing Problem.*

Figure 1 shows three views of a section of a routing channel in a symmetrical FPGA (note that, for clarity, the vertical channels are not shown in the picture). In each view, the figure illustrates the routing options that are available in this channel for three different connections, called A, B, and C. In the figure, a *wire segment* in the channel is shown as a solid horizontal line, and a wire segment that is available for a particular connection is highlighted as a bold line. A routing switch that joins two horizontal wire segments is drawn as a dashed line, and a switch that joins a horizontal segment to a logic block (L) pin is shown as an X. Finally, logic block *pins* are drawn as vertical lines. As indicated in Figure 1, the routing architecture in this FPGA has three tracks and the routing switches are distributed such that only tracks 2 and 3 can connect the required logic block pins for Connection A, and only tracks 1 and 2 can be used for Connections B and C. The discussion below considers this routing problem, first from the perspective of just completing all three connections, and then also considering the usage of the wire segments according to their lengths.

Assume that a detailed router completes connection A first. If it chooses to route Connection A on track 2, then one of B and C will fail because they both rely on the single remaining option, namely track 1. On the other hand, if the router had chosen track 3 for A, then B could use track 1 and C track 2, or vice-versa. This simple example illustrates that, even when there are only three connections involved, routing decisions made for one connection can unnecessarily block others. This is the main reason why detailed routing for FPGAs is more difficult than classical detailed routing.

The above routing solution satisfies the goal of completing all three connections, but only one of the two choices for B and C makes the best use of the available wire segments. Specifically, it is clear from examining the routing channels that Connection B should be assigned to track 2, since the wire segment there exactly matches the connection's length. This also leads to the best solution for Connection C since it requires only one wire segment in track 1 but would need two segments in track 2. Matching the lengths of wire segments to connections is a new problem that does not exist for classical mask-programmed technologies, where there is complete flexibility to create metal wires of any length. While Figure 1 shows only connections within one small routing channel, the problem is much more complex where many connections compete for wire segments and when both horizontal and vertical channels are considered.

Common approaches used for detailed routing in other types of devices are not suitable for FPGAs. Classic Maze routing [9] is ineffective because it is inherently sequential and so, when routing one connection, it cannot consider the side-effects on other connections. The example in Figure 1 illustrates why this is important. Channel routers [10] are not appropriate for symmetrical FPGAs because it is very difficult to subdivided the routing problem into independent channels. Channel routing algorithms are used in [11] and [12] for row-based FPGAs [13] [14]. This is possible for these types of FPGAs because the logic blocks are arranged in rows separated by routing channels and the routing switches are such that each logic block pin can connect to all the wire segments in the channels above and below it and each horizontal wire segment can connect to all the vertical wire segments that cross it. This routing flexibility is impractical in symmetrical FPGAs, because too many routing switches would be required.

There have been some previous publications concerning detailed routing for symmetrical FPGAs. The earliest one [15] [16] can be considered to be the predecessor to the SEGA algorithm described in this paper. The algorithm in [16] addresses the problem of considering the side-effects that routing one connection has on others. However, the algorithm was intended for routing architectures that consisted of short wire segments only, so it does not have the ability to match the lengths of wire segments to the lengths of connections. Section 5 of this paper compares routing results for the algorithm in [16] to those produced by SEGA. Alternative approaches to routing in symmetrical FPGAs can be found in [17] and [18]. No direct

comparison is available between these algorithms and SEGA, although such a comparison would be of interest.

This paper is organized as follows: Section 2 gives the model of the FPGA supported by the router, Section 3 defines the general approach used to solve the routing problem, Section 4 describes the SEGA routing algorithm, Section 5 gives results showing the performance of SEGA on a suite of benchmark circuits, and Section 6 provides concluding remarks.

2 The FPGA Model

The model for FPGAs supported by SEGA is similar to that used in earlier papers on FPGA architecture [19] [20] [21] and CAD algorithms [15] [16]. As illustrated in Figure 2, the FPGA consists of a rectangular array of logic blocks with *N x M* blocks, and both horizontal and vertical routing channels. In terms of commercially available devices, the structure depicted in the figure is most similar to that found in Xilinx FPGAs [1] [2] [3], but it is more general. The FPGA in Figure 2 has two pins on each side of a logic block (L) and three tracks per channel. No assumptions are necessary about the internal details of the logic blocks, except that each block has some number of pins that are connected to the channels by routing switches. The channels comprise two kinds of blocks, called Switch (S) and Connection (C) blocks, described below. The S blocks hold routing switches that can connect one wire segment to another, and the C blocks house the switches that connect the wire segments to the logic block pins. Note that the blocks are numbered along the left and bottom sides. These coordinates are referenced later as a means of describing connections to be routed.

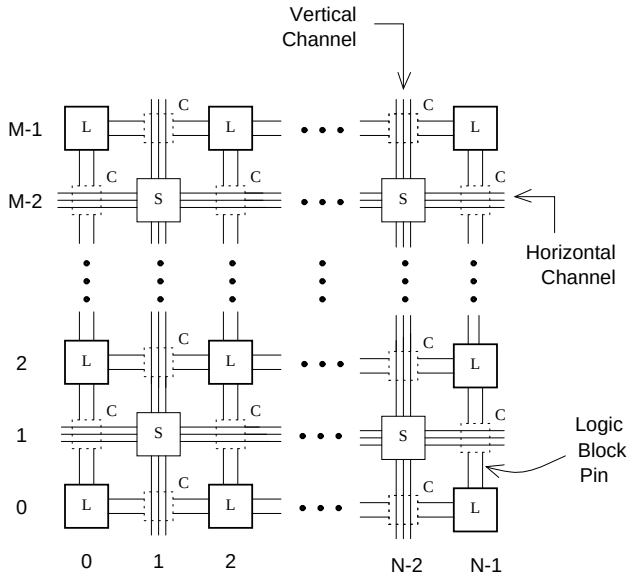

Figure 2 - *An N* x *M FPGA.*

The general nature of the S block is illustrated in Figure 3a. Notice that, since segmented routing channels are supported, some tracks pass straight through the S block, while other tracks are broken by routing switches. There are two representations for routing switches in the figure, either as a dotted line for connecting the ends of two wire segments, or as an X for a wire segment that passes straight through the S block. In the figure, the S block switches allow the horizontal tracks numbered 1, 2, and 3 to connect to the vertical tracks with the same numbers. Although Figure 3a provides a specific example of an S block, the

SEGA router treats the S block as a general four-sided switch block that can be configured in any way according to the routing architecture of the FPGA.

There are two parameters that determine the configuration of the routing switches in an S block. The first is the segmentation of the channels; by allowing customizing of the S blocks, SEGA supports virtually any channel segmentation. In the current implementation of the router, the channel segmentation is selected by a user-specified probability distribution. The distribution is used to create wire segments of a certain length with an appropriate frequency. The second parameter, called the *flexibility* of the S block, is set by a parameter F_s, which defines the number of other wire segments that a wire segment that ends at an S block can connect to. For the example shown in Figure 3a, the wire segment at the top left of the S block can connect to three others and so F_s is 3. Note that F_s alone does not determine the number of routing switches in the S block, since tracks that pass uninterrupted through the S block have fewer associated switches.

Figure 3b illustrates a C block. The tracks pass uninterrupted through the C block and can be connected to the logic block pins via the set of switches. The flexibility of a C block, F_c, is defined as the number of wire segments in the C block that each logic block pin can connect to. For the example shown in the figure each pin can be connected to 2 vertical tracks, and so F_c is 2 (in the C block, a routing switch is drawn as an X). SEGA allows complete customizing of the C block, depending on the FPGA's architecture.

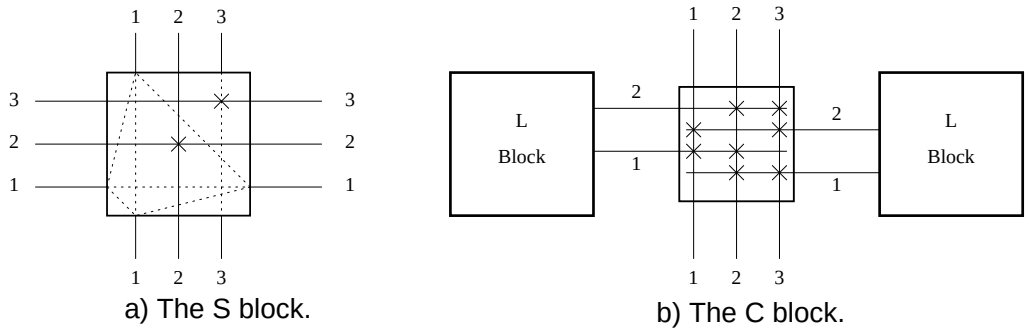

Figure 3 - *Examples of S and C Blocks.*

The main advantage provided by the FPGA model described above is its generality. A wide range of routing architectures can be represented by changing the number of tracks per channel and the contents of the C and S blocks. This means that SEGA can be used as a tool for investigating the effects of an FPGA's routing architecture on the routability of circuits. Previous research [19] [20] [21] has already examined the effects of the F_c and F_s parameters, but much work is still needed with regard to channel segmentation.

3 General Approach and Problem Definition

As for other VLSI technologies, FPGA routing problems have a high degree of combinatorial complexity. The general approach used here is the standard [8] two-stage method of global routing followed by detailed routing. This allows the separation of two distinct problems: balancing the densities of the routing chan-

nels, and assigning specific wire segments to each connection. The global router employed is an adaptation of the LocusRoute global routing algorithm for standard cells [22]. This global router divides multi-point nets into two-point connections and finds minimum distance paths through the routing channels for each connection. The global routing algorithm's main goals are to distribute the connections among the channels so that the channel densities are balanced, and to minimize the number of "turns" for each connection. Intuitively, these are sensible goals for FPGAs, because the capacity of each channel is strictly limited, and because connections can make better use of long segments if they do not turn corners.

The function of the global router is to define a coarse route for each connection by assigning it a sequence of channel segments. Figure 4a shows a representation of a typical global route for one

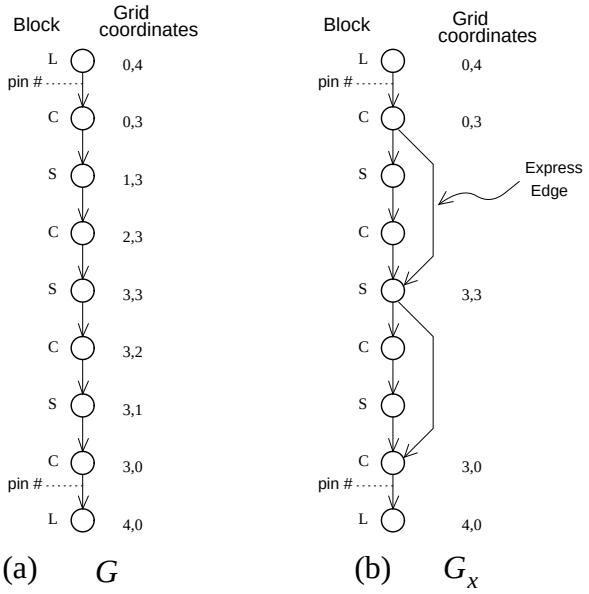

Figure 4 - *(a) A Coarse Graph and (b) the Coarse Graph with Express Edges.*

connection. The global route is called a *coarse graph*, *G*, where the L block at (0,4) is referred to as the root of the graph and the L block at (4,0) is called the leaf. The vertices and edges of *G* are identified by the coordinates shown in Figure 2. The root vertex in *G* specifies that the connection begins at the logic block at coordinates (0,4). The edge attached to the root would be labelled to reference a specific pin on that logic block. Similarly, the leaf vertex shows that the connection ends at the logic block at (4,0), and the last edge would again refer to a specific pin number. The other vertices and edges define a sequence of channel segments that the global router has chosen to connect the logic block at location (0,4) to the one at (4,0).

Since the global router splits all nets into two-point connections, the coarse graphs always have a fan-out of one. This implies that some connections that are part of the same net will overlap within a routing channel. It is possible to merge the wire segments assigned to such connections during routing to minimize wastage due to this issue.

4 The SEGA Detailed Routing Algorithm

The SEGA routing algorithm has been implemented in a program, written in the C programming language. After creating a representation of the FPGA, from a set of user-specified parameters, the program inputs

the netlist that describes the circuit to be routed (i.e. the output of the global router) and a coarse graph is created for each required connection. Detailed routing then proceeds in two main phases: in phase 1, the router examines the wire segments and routing switches present in the FPGA and enumerates all of the alternatives for the detailed route of each coarse graph. Then, in phase 2, viewing all the alternatives at once, specific routing decisions are made for each connection. The decisions made in phase 2 are driven by a four-part cost function that is evaluated based on the alternatives enumerated in phase 1.

4.1 Phase 1: Enumerating the Detailed Routes

During phase 1, SEGA enumerates all of the detailed routes that are available in the FPGA to implement each coarse graph. Since one of the main focuses of SEGA is to match the lengths of wire segments to the lengths of connections, the first step is to determine how far each connection travels along a single channel before turning onto another channel. When forming connections in the next phase, the router will then be able to match wire segment lengths to each straight portion of a connection. The information is recorded in the coarse graphs, as illustrated in Figure 4b. The figure shows a modified coarse graph, called G_x, where the extra edges that are shown correspond to the straight portions of the coarse graph. As indicated in Figure 4b, the extra edges in G_x are called "express edges," because they show how far the connection travels without turning. The next step for the algorithm is to examine the routing resources in the FPGA to determine the detailed routes that are available to implement each express edge, as explained below.

The alternative detailed routes for each G_x can be represented in an *expanded* graph, called *D*. Each edge in *D* represents specific wire segments (one or more) that can be used to implement the corresponding express edge in G_x. The example shown in Figure 4 illustrates an expanded graph corresponding to G_x

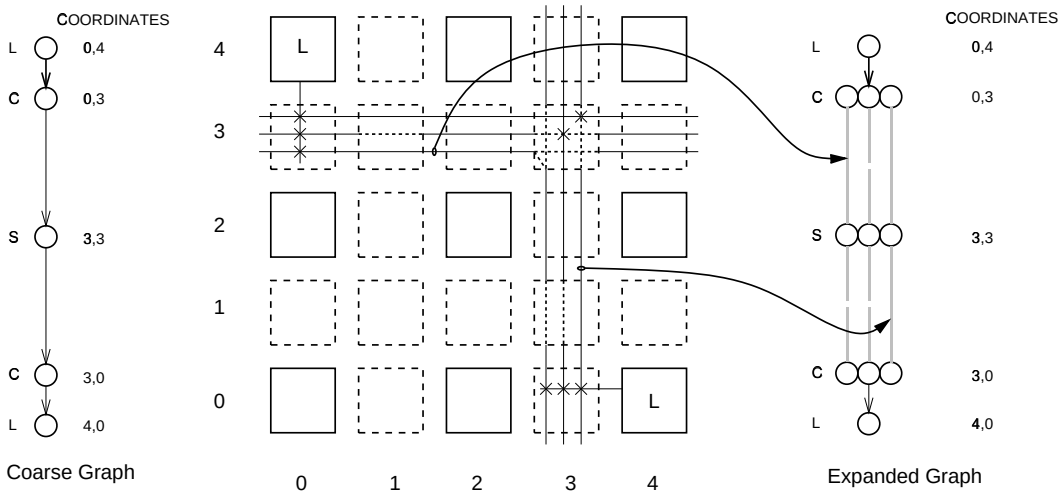

Figure 5 - *D, Showing the Alternative Detailed Routes for* G_x.

from Figure 4b. Note that only the express edges of G_x are shown in the figure. Figure 4 also includes the details of the routing channels in the FPGA that are involved for the connection. As shown, *D* has the same vertices as G_x, but there is one instance of each vertex for each *path* in the FPGA that leads from the root vertex to the leaf vertex. The edges of *D* are drawn as shaded lines to indicate that they are not simple

edges. Each edge, *e*, may imply the use of multiple wire segments, in which case multiple shaded lines are shown. It is important to realize that the length of a wire segment referenced in *e* is not necessarily the same as the length of the corresponding express edge in G_x, since a wire segment may be either longer or shorter than the express edge itself. Each *e* has associated with it one or more labels, one for each wire segment that it references. The label identifies the corresponding wire segment in the FPGA, examples of which are shown by the two curved lines pointing from edges in *D* to wire segments in the FPGA.

4.2 Phase 2: Connection Formation

After phase 1, each *D* may contain a number of alternative paths. SEGA places all the paths from all the expanded graphs into a single path list. Based on a four-part cost function, the router then selects paths from the list. Each selected path defines the detailed route of its corresponding connection. Because SEGA expands all the coarse graphs before making any routing decisions, it can consider the side effects that a decision made for one connection has on others. For reasons given earlier, this is important in FPGAs. In addition, the express edges in each G_x and the multiple wire-segment edges in each *D* allow the router to match the straight sections of a connection with appropriately long wire segments. Phase 2 proceeds as follows (the *cost* of a path will be defined shortly):

put all the paths in each expanded graph into the path list
while the path list is not empty {
    select the path with the lowest cost
    mark the graph corresponding to the selected path as routed, and remove all paths in this graph from the path list
    find all paths that would conflict with the selected path (i.e. all paths that are part of *different* nets but reference the wire segments just allocated to the selected path) and remove them from the path list. If a connection loses its last remaining path, that connection is deemed unroutable[1]
    update the cost of all affected paths
}

4.2.1 Cost Function Design

Each path, *p*, in an expanded graph has a cost composed of four binary-weighted terms:

$$Cost = w_\alpha C_\alpha(p) + w_\beta C_\beta(p) + w_c C_c(p) + w_f C_f(p) \quad (1)$$

The first two terms concern the allocation of wire segments according to the lengths of connections, and the last two focus on the task of successfully routing all connections. The binary weights allow each term to be “turned on/off” to evaluate its effectiveness. Each cost term is defined further below:

1. Note that it would be desirable for the router to have some means of trying other alternative solutions when a connection fails to route. For example, the router could perform another iteration on the problem, trying different combinations of the cost function terms for the channels that contain unrouted connections. Also, the router could try a different global route for the failed connection. These features are not yet implemented in SEGA, but the routing quality is already excellent as shown in Section 5.

$C_\alpha(p)$ — This cost is similar to one described in [12]. Its purpose is to minimize the wastage of long wire segments by short connections. Thus, $C_\alpha(p)$ is defined as the quotient of the total wasted length of the segments in p divided by the total length of the segments in p.

$C_\beta(p)$ — This term is similar to a cost used in [11] and [12]. Its purpose is to minimize the number of segments used. Thus, $C_\beta(p)$ is defined as the quotient of (the actual number of segments in p minus the minimum possible) divided by (the actual number of segments in p).

$C_c(p)$ — The purpose of this term is to allow the router to select whichever connection has the fewest routing alternatives remaining. $C_c(p)$ is defined as the ratio of the number of remaining paths in an expanded graph to F_c, the maximum number of paths in any expanded graph[1].

$C_f(p)$ — This cost was originally defined in [15] [16]. Its purpose is to allow the router to select a path to route such that it has the least negative effect on other connections. $C_f(p)$ is calculated by keeping track of all occurrences in expanded graphs of every wire segment in the FPGA. $C_f(p)$ is defined as a summation of the *demand* for each edge in p. To calculate the demand for an edge, e, SEGA counts the number of instances of wire segments in e that are in other expanded graphs. However, some instances are less likely to be used because there are alternatives edges in parallel with e. Thus, if p contains a total of i edges, called (e_1, e_2, $\ldots e_i$), and each edge has j other occurrences (e_1, e_2, $\ldots e_j$), then $C_f(p)$ is given by

$$C_f(p) = \sum_i \sum_j \frac{1}{alt(e_j)} \qquad 2$$

where $alt(e_j)$ is the number of edges in parallel with e_j.

In the current implementation of SEGA, the $C_f(p)$ cost is not added to the others as shown in Equation 1, but rather the other three costs are added and $C_f(p)$ is used to break ties. From the point of view of obtaining a good quality result, this arrangement makes intuitive sense because $C_c(p)$ allows the router to prioritize connections with fewer choices, and $C_f(p)$ shows which of the choices for that connection has the least negative effect on others.

5 Results

This section contains two types of results concerning the effectiveness of the routing algorithm. The first type shows that the algorithm achieves excellent quality in terms of routing completion. The second result provides a measure of how well SEGA performs at matching the lengths of connections to the lengths of wire segments. In both cases, the routing architecture used is similar in terms of connectivity to the Xilinx

1. F_c is the maximum number of paths only if the expanded graphs have a fanout of one. This is true for values of F_s that do not exceed three. Research has shown that values of F_s greater than three offer no real benefit [19] [20] [21].

XC4000 series FPGAs. That is, F_s is three and F_c is equal to *W,* where *W* is the number of tracks per routing channel.

5.1 Routing Completion Results

For these results, the FPGA was configured so that the channels consisted of only short wire segments. This allows the SEGA router to be directly compared with an earlier router for symmetrical FPGAs, called CGE [16]. For this experiment, a set of ten benchmark circuits was routed using both SEGA and CGE. The circuits used for the experiment are mostly from an MCNC benchmark suite and range in size from 202 to 2135 two-point connections

The familiar yardstick of channel density is used as a measure of the quality of the routing results. The "Channel Density" column in Table 1 shows the maximum channel density over all channels for each cir-

Circuit Name	Channel Density	Min. W for SEGA	Min. W for CGE	Min. W for 'Maze'
alu4	13	15	15	20
apex7	13	13	13	15
term1	9	10	10	13
z03	14	14	14	21
example2	17	17	18	21
too_large	11	12	13	17
k2	15	17	19	27
vda	13	13	14	19
9symml	10	10	10	12
alu2	10	11	12	17
Total	125	132	138	182

Table 1 - *Characteristics of Benchmark Circuits.*

cuit. This is a lower-bound on the required number of tracks per routing channel. The table gives the minimum number of tracks needed to route 100 percent of the connections in each circuit for both SEGA and CGE. As shown, the results are similar but SEGA uses 6 tracks less in total for the ten circuits. Examination of Table 1 also shows that averaged over the ten circuits, SEGA requires only 0.6 tracks above the channel density, which is excellent. The far right hand column of Table 1 lists the minimum number of tracks needed for an algorithm that does not consider the side-effects that one connection has on others. For this column, the cost function that is normally used by the CGE router was disabled. This turns the algorithm into a sequential router, similar to a classical maze router. For this case, a total of 38 percent more tracks per channel are required above the results achieved by SEGA. This shows that it is important to consider the side-effects that routing decisions for one connection have on others, and that SEGA addresses this issue well.

5.2 Wire Segment Allocation Results

For these results, the FPGA has been configured with segmented tracks. The segment lengths are determined by a Poisson distribution with mean 0.5, resulting approximately in the following distribution of wire segment lengths: 60% length 1, 30% length 2, 8% length 3, and 2% longer than length 3. Table 2 shows, with this segmentation, the effectiveness of the C_α and C_β cost terms. These results are based on

Cost Term Used	Circuit Name and Segment Statistics			
	vda		9symml	
	# of segments (min. segs. = 1622)	length of segments (min. len. = 3304)	# of segments (min. segs. = 566)	length of segments (min. len. = 888)
$w_\alpha = 0, w_\beta = 0$	2496	3758	665	1052
$w_\alpha = 1, w_\beta = 0$	2556	3425	681	941
$w_\alpha = 0, w_\beta = 1$	2263	3828	619	1053
$w_\alpha = 1, w_\beta = 1$	2321	3475	626	969

Table 2 - *Segment Allocation Results.*

two of the benchmark circuits[1]. For each circuit, the table lists the "Minimum Segments" that could be used for the circuit (this is set by the total number of express edges in all of the coarse graphs) and the minimum total length of segments (this is given by the total number of wire segments that would result if only length one segments were used for all portions of all connections). The table then shows, as indicated by the "Cost Term Used" column, four experiments to test the performance of SEGA at allocating wire segments according to the length of connections. In the table, the meaning of $w_\alpha = 0, w_\beta = 0$ is that neither the C_α or C_β costs are used for that line of the table. Similarly, $w_\alpha = 1, w_\beta = 1$ means that both of the cost terms are used.

Table 2 shows that when neither of the cost terms are utilized, both the total number of segments and the total length of segments used is high. However, when C_β is used, the total number of segments allocated decreases by about 10%. Similarly, when the C_α term is utilized, the total length of segments used decreases by about 10%. Finally, as the bottom row of the table shows, when both C_β and C_α are employed, the segment allocation performs well by both measures. Table 3 also gives the minimum possible number of segments and segment lengths for each circuit, but these are based only on the global routes assigned to the connections which does not consider the segmentation of the FPGA's routing tracks.

1. These experiments have been carried out for the other circuits as well and show similar results.

6 Conclusions

This paper has described a new kind of detailed routing algorithm for symmetrical FPGAs with segmented routing channels. The algorithm achieves an excellent result in terms of the number of tracks per channel needed to route a given circuit, and in terms of the allocation of wire segments according to lengths of connections. For this latter metric, the router performs well both at limiting the number of wire segments allocated for long connections and the lengths of segments assigned to short connections. In future work, the algorithm will be enhanced by adding an iterative procedure to assist in routing difficult connections.

References

[1] W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo and S. L. Sze, "A User Programmable Reconfigurable Gate Array," *Proc. 1986 Custom Integrated Circuits Conference*, May 1986, pp. 233-235.

[2] H. Hsieh, K. Duong, J. Ja, R. Kanazawa, L. Ngo, L. Tinkey, W. Carter and R. Freeman, "A Second Generation User-Programmable Gate Array," *Proc. 1987 Custom Integrated Circuits Conference, May 1987*, pp. 515 - 521.

[3] H. Hsieh, W. Carter, J. Ja, E. Cheung, S. Schreifels, C. Erickson, P. Freidin, L. Tinkey and R. Kanazawa, "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays," *Proc. 1990 Custom Integrated Circuits Conference*, May 1990, pp. 31.2.1 - 31.2.7.

[4] Quicklogic, "An Introduction to Quicklogic's pASIC Devices and SpDE Development Environment," *Data Sheet from Quicklogic*, April 1991.

[5] ALTERA Corporation, "FLEX Programmable Logic," *Product Information Bulletin*, September 1992.

[6] Stephen D. Brown, Robert J. Francis, Jonathan Rose and Zvonko G. Vranesic, "Field-Programmable Gate Arrays," Kluwer Academic Publishers, 222 pages, 1992.

[7] J. Soukup, "Circuit Layout," *Proc. of the IEEE*, Vol. 69, No. 10, pp. 1281-1304, October 1981.

[8] Chapter 5 of "Physical Design Automation of VLSI Systems," B. Preas and M. Lorenzetti, Ed., Benjamin/Cummings.

[9] C. Lee, "An algorithm for path connections and its applications," *IEEE Transactions on Electronic Computers*, VEC-10, pp. 346-365, Sept. 1961.

[10] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," *Proc. 8th Design Automation Conference*, pp. 155-163, 1971.

[11] J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented Channel Routing," *Proc. 27th Design Automation Conference*, pp. 567-572, June 1990.

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

[13] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat and A. Mohsen, "An Architecture for Electrically Configurable Gate Arrays," *IEEE Journal of Solid State Circuits*, Vol. 24, No. 2, April 1989, pp. 394-398.

[14] M. Ahrens, A. El Gamal, D. Galbraith, J. Greene, S. Kaptanoglu, K. Dharmarajan, L. Hutchings, S. Ku, P. McGibney, J. McGowan, A. Samie, K. Shaw, N. Stiawalt, T. Whitney, T. Wong, W. Wong and B. Wu, "An FPGA Family Optimized for High Densities and Reduced Routing Delay," *Proc. 1990 Custom Integrated Circuits Conference*, May 1990, pp. 31.5.1 - 31.5.4.

[15] S. Brown, J. Rose and Z. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays," *Proc. IEEE International Conference on Computer Aided Design*, pp. 382-385, Nov. 1990.

[16] S. Brown, J. Rose and Z. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 11, No. 5, pp. 620-628, May 1992.

[17] Mikael Palczewski, "Plane Parallel A* Maze Router," *29th ACM/IEEE DAC*, pp. 691-697, June 1992.

[18] Jon Frankle, "Iterative and Adaptive Slack Allocation for Performance-driven Layout and FPGA Routing," 29th ACM/IEEE DAC, pp. 536-542, June 1992.

[19] J. Rose and S. Brown, "The Effect of Switch Box Flexibility on Routability of Field-Programmable Gate Arrays," *Proc. 1990 Custom Integrated Circuits Conference*, pp. 27.5.1-27.5.4, May 1990.

[20] J. Rose and S. Brown, "Flexibility of Interconnection Structures in Field-Programmable Gate Arrays," *IEEE Journal of Solid State Circuits*, Vol. 26 No. 3, pp. 277-282, March 1991.

[21] B. Tseng, J. Rose and S. Brown, "Using Architectural and CAD Interactions to Improve FPGA Routing Architectures," *First International ACM/SIGDA Workshop on Field-Programmable Gate Arrays*, pp. 3-8, February 1992.

[22] J. Rose, "Parallel Global Routing for Standard Cells," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 10, pp. 1085-1095, Oct. 1990.