Minimizing FPGA Interconnect Delays

STEPHEN BROWN
MUHAMMAD KHELLAH
ZVONKO VRANESIC
University of Toronto

Optimizing FPGA routing architectures for speed performance also involves improving the CAD tools for mapping circuits. Although their results are sensitive to the tools used, the authors draw several basic conclusions about both FPGA routing architectures and CAD tools.

USING SRAM-BASED field-programmable gate arrays is desirable because of their relative low cost (compared to traditional gate arrays) and reprogrammability. The added versatility, however, comes with logic capacity and speed performance considerably lower than that of gate arrays. This is because of the added chip area and capacitive loading of programmable switches. Our work considers ways to improve the area and speed performance of SRAM-based FPGAs by enhancing their interconnect.

Researchers have studied the speed performance of FPGA routing architectures,[1,2] but only for antifuse-based FPGAs similar to those offered by Actel. The issues associated with Xilinx-style SRAM-based devices are different from those in Actel-style FPGAs, because the general structures of the chips are dissimilar. In addition, the delay characteristics of pass transistors and antifuses are quite different.

Early SRAM-based FPGAs[3] featured routing channels that consisted mostly of short wire segments. Joining two or more segments via routing switches formed longer segments. Long connections passing through several switches in series, however, produce a large delay. This occurs because pass-transistor switches have significant series resistance and parasitic capacitance. More recent SRAM-based products[4] include wire segments of various lengths, but no previous study has investigated the trade-offs in selecting a particular routing architecture scheme. Since the wire segment lengths available in an FPGA are key to speed performance, this is the central theme of the research reported here.

FPGA interconnect architecture

We provide a detailed example of how to design FPGA architectures by examining several important issues associated with interconnect resources for FPGAs that use SRAM programming technology. Our experiments examine two important metrics: the speed performance of implemented circuits and the effective use of available interconnect resources. The goal is to improve upon FPGA speed performance by decreasing delays associated with the interconnect. Our results are most directly applicable to FPGA architectures similar in style to the Xilinx XC4000 series. However, some significant results are of a more general nature and perhaps applicable to other styles of FPGAs as well.

In addition to routing architectures, we address the CAD tools that allocate these routing resources to implement circuits. The suite of CAD tools we used was custom developed for FPGA architecture experiments. Its developers parameterized these tools to work with an FPGA model with many variable parameters. The key CAD tool used in

0740-7475/96/$05.00 © 1996 IEEE

this study is the Sega (Segment Allocator) detailed router,[5] which has parameters that allow virtually any mixture of wire segment lengths in the routing channels.

Researchers have developed several other excellent detailed routers recently. PathFinder[6] uses an approach that trades off routability and speed performance through multiple iterations. Although PathFinder supports most FPGA routing architectures, it requires a graph to describe an architecture, and is thus inconvenient for experimentation over a wide range of wire segment lengths. Tracer-FPGA,[7] Orthogonal Greedy Coupling,[8] and the Steiner Tree[9] algorithms achieve excellent results as measured by the number of tracks needed to route benchmark circuits. However, all of these algorithms assume that the FPGA contains wire segments of only one length, and so are of no use for experimentation on segment length.

FPGA model. The style of FPGA architecture that this study assumes is more similar to Xilinx devices than to other products. This model is similar to that employed in other studies on FPGA architecture and CAD algorithms, with routing channels in vertical and horizontal tracks. Each track contains wire segments that can be of any length, where the length of a segment is the number of logic blocks it spans. As an illustration, Figure 1 shows a section of a horizontal channel with wire segments of lengths 1, 2, and 3. Wire segments that abut can be joined through a switch; the figure does not show these switches. For simplicity, the figure does not show the vertical channels.

A key characteristic of the FPGA model is that the channels comprise two kinds of blocks, switch (S) and connection (C), characterized by parameters *Fs* and *Fc*. For our experiments, these parameters are $Fs = 3$, and $Fc = W$, where W is the number of tracks in each channel. Recent publications on detailed routing algorithms[7-9] also use these assumptions, indicating their general acceptance as reasonable.

Experimental procedure. We used several CAD tools that would be available in a typical FPGA development system. The benchmark circuits come from the Microelectronics Center of North Carolina (MCNC) logic synthesis benchmark suite. We technology mapped the circuits, placed logic blocks with a min-cut algorithm implementation, and performed routing using the traditional two-stage method of global routing followed by detailed routing.

Figure 2 illustrates the overall flow of the CAD system. Since this article focuses on routing issues, we performed the CAD stages before routing only once for each benchmark circuit. The diamond shapes on either side of Figure 2 represent our experimental procedure in which the FPGA routing architecture and parameters of the routing CAD tools are variables that take on specific values for a particular experiment. We evaluate the results by asking two questions after the detailed-routing stage: Did the routing tools successfully complete all of the required circuit connections? What is the speed performance of the final result?

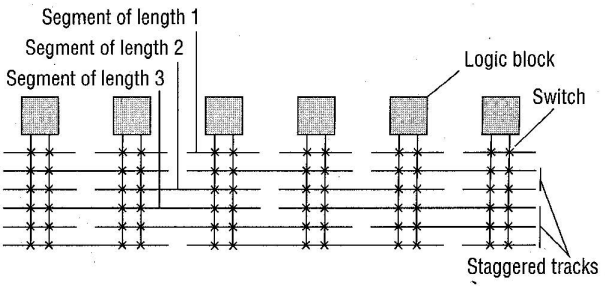


Figure 1. *Channel with wire segment lengths of 1, 2, and 3.*

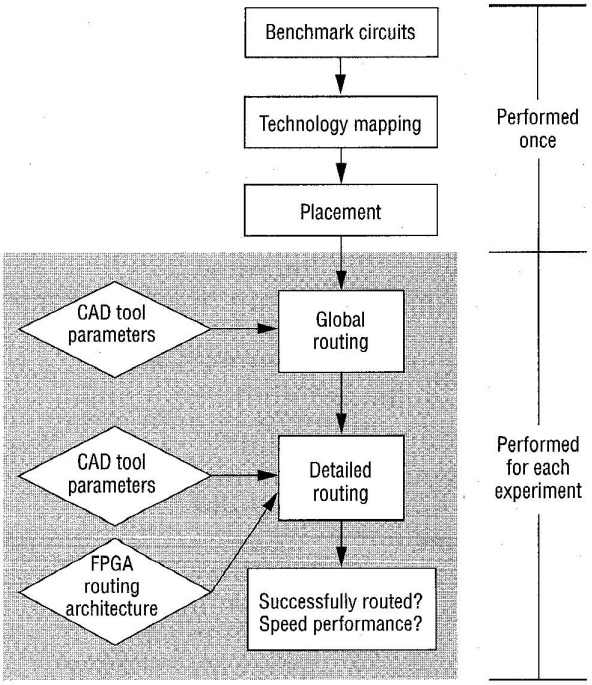


Figure 2. *CAD system overview.*

We can answer the first question after running the routing tools. To answer the second question, we estimate routing delays of signals using the method described later in this section. The following subsections provide more details on the global- and detailed-routing stages.

Global router. We employed an adaptation of a global-routing algorithm for standard cells. Based on point-to-point con-

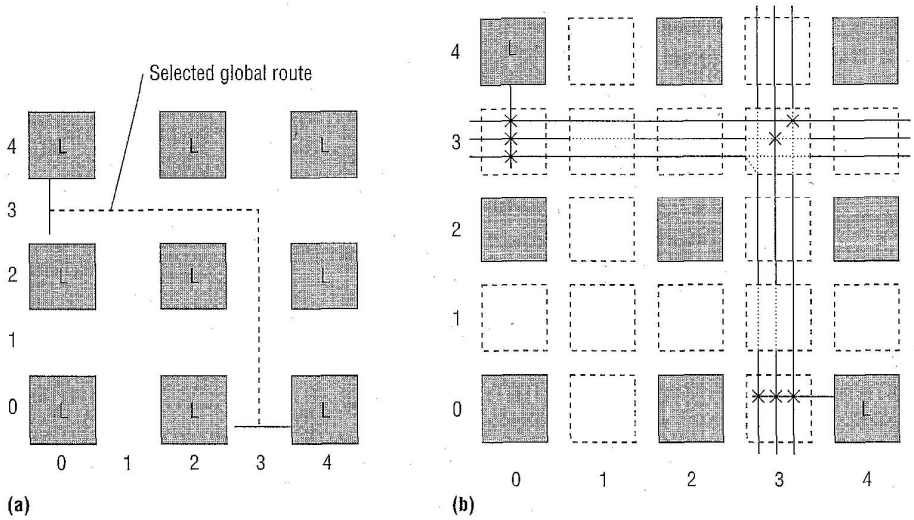


Figure 3. *Global route for a connection* ***(a)*** *and corresponding possible detailed routes* ***(b)***.

Table 1. *Sega's cost functions.*

Cost function	Description
Completion	Optimizes for routability only
Seg_Len	Minimizes lengths of wire segments
Num_Seg	Minimizes the number of wire segments
Seg_Len + Num_Seg	Combines the second and third cost functions
Analytic_Model	Uses an analytic model to find actual delays
Net_Routing	Optimizes for speed, but also focuses on reuse of wire segments for connections on same net

nections (as opposed to multipoint nets), the router divides multipoint nets into a set of two-point connections as its first step. (We discuss the implications of this step later.) It then finds minimum-distance paths through the routing channels for each connection. The algorithm's main goal is to distribute connections among the channels to balance the channel densities. Intuitively, this is a sensible goal for FPGAs because the capacity of each channel is strictly limited.

Figure 3a shows an example of global routing in which a logic block pin at coordinates (0,4) is to connect to a pin at (4,0). Although we could use other global routes through the channels for this connection, the router has selected the one indicated by the dashed line. To describe its solution to the detailed router that will later complete the connection, the router produces a set of coordinates recorded in a graph that marks the global route. It does this for every circuit connection.

Detailed router. The Sega[5] detailed-routing algorithm developed for this study specifically handles array-based FPGAs with segmented channels. Sega can produce a routing result optimized either for the best routing completion (minimizing the amount of routing resources and hence, chip area) or the best speed performance of the implemented circuit. Its developers parameterized Sega to support any FPGA architecture that has both vertical and horizontal tracks.

Sega input is a netlist of two-point connections—the output of the global router. As an example, Figure 3b shows a set of detailed routes that might be available to realize the global route in Figure 3a. To route connections, Sega allocates wire segments according to a cost function, basing its decisions on optimization either for routing completion or for speed. For the former, it only considers the circuit routability, which means the router focuses only on successfully routing 100% of the circuit connections. In the second mode, Sega selects the routes with the best speed performance.

As a research vehicle, Sega can use one of several cost functions to assess the speed performance of a detailed route. Table 1 lists the cost functions and indicates their optimization goals; Khellah provides further details on these functions.[10] To illustrate the impact of the detailed routing algorithm on the FPGA's speed performance, a later section compares routing results produced by each cost function in Table 1. Since the measurement of timing delays in routed connections for SRAM-based FPGAs is not obvious, we describe the method used.

Delay model for estimating speed performance. Two reasons make it necessary to measure the propagation delays of routed connections in FPGAs. First and most importantly, once we have fully implemented a circuit, we need to measure its speed performance. Second, when using the Analytic_Model cost function (Table 1), the detailed router calculates actual routing delays of alternative paths to make

routing decisions. For both these situations, we need an efficient analytic modeling technique to quickly and accurately estimate signal propagation delays.

To estimate routing delays in FPGAs, we have developed an adaptation of the analytic modeling technique presented by Rubinstein, Penfield, and Horowitz,[11] which models MOS transistors as constant RC elements. Although the original publication[11] stated that the model was not applicable to pass transistors, we showed that the model can accurately represent pass transistors with a careful choice of resistance and capacitance values.[10] The input to the analytic model is an RC tree in which resistors represent routing switches that signals pass through in series. Capacitors correspond to parasitic capacitance from both routing switches and wire segments. As output, the model produces an estimate for the delay from the network source node to each sink node. The time it takes an ideal step input at the source to reach half its value at the sink is the source-to-sink delay.

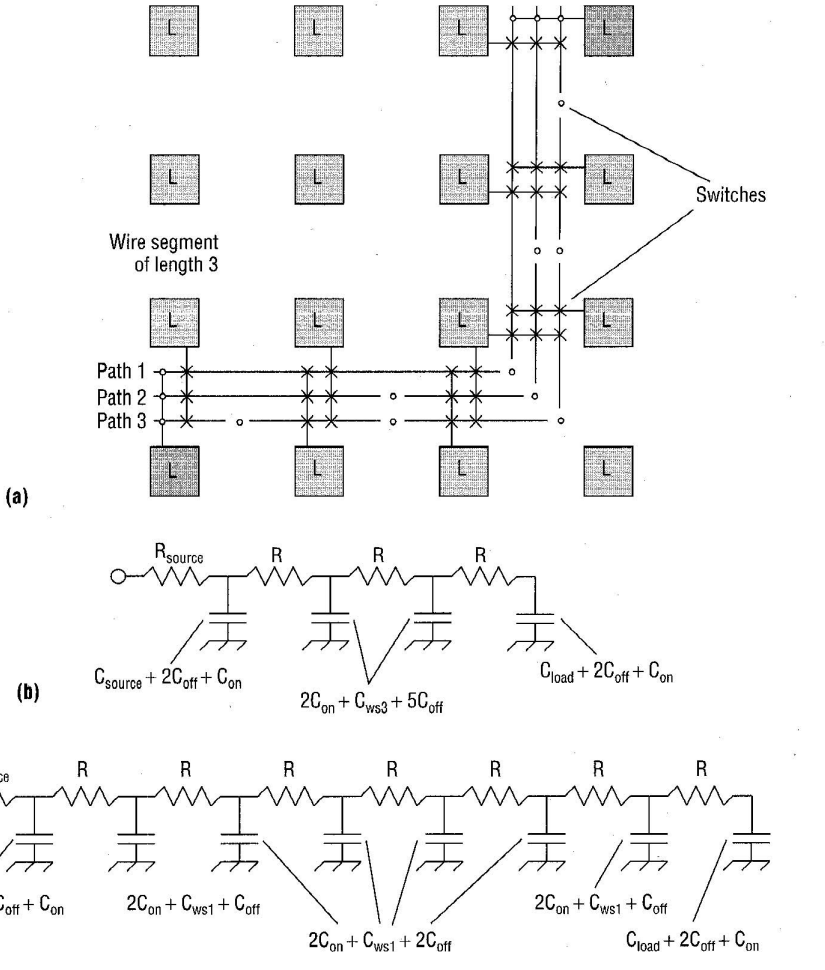


Figure 4. *Alternative routing choices for a net* ***(a)*** *and example RC trees for path 1* ***(b)*** *and path 3* ***(c)*** *routed in an FPGA.*

In Figure 4, two example RC trees for detailed routes (labeled paths 1 and 3) connect the L block at the figure's lower left to the block at the upper right. Each routing switch in series along a path (denoted by a small circle) contributes a resistor and capacitor to the RC tree. Routing switches for connections to other L block pins and wire segments themselves add capacitance. (In the figure, we refer to this capacitance as C_{off} for switches that are off, and C_{ws1} and C_{ws3} for wire segments of lengths 1 and 3.) Each net includes a source resistance and capacitance as well as a load capacitance.

For the results presented later, we calculated R and C assuming a 0.8-micron BiCMOS process, letting R_{on} equal 915 ohms, C_{on} equal 25 fF, C_{off} equal 13 fF, C_{ws1} equal 3 fF, and C_{ws3} equal 9 fF.[10] With these parameters, the analytic model calculates the speed performance of individual nets directly. We then define the delay of a net as the largest delay from the net's source to any of its sinks. The speed performance of an entire circuit implemented in an FPGA is the average of the net delays in the circuit.

We use net delay as a speed performance measure because the goal of the experimental procedure is to evaluate the speed performance provided by an FPGA routing architecture. If we had meant our experiments to measure the maximum achievable speed performance of a specific circuit, then average net delay would not be the most appropriate measure; it would be better to find the delay of the circuit's critical path.

Results. We based our experiments on the set of benchmark circuits summarized in Table 2 (next page). All the circuits except the largest are from the MCNC benchmark suite.

Table 2. Benchmark circuit characteristics.

Circuit	No. of logic blocks	No. of multipoint nets	No. of two-point connections
9symml	72	79	259
too_large	156	186	519
apex7	80	126	300
example2	120	205	444
vda	210	225	722
alu2	143	153	511
alu4	255	256	851
term1	56	88	202
C1355	110	145	360
C499	110	145	360
C880	120	174	427
k2	360	404	1,256
z03D4	586	608	2,135

(The largest circuit was from Bell Northern Research.)

Effect of CAD routing tools. We first considered the speed performance and area efficiency issues associated with the detailed router by varying its parameters and measuring their effects on the circuit implementations. These experiments illustrated that the CAD tools used have a marked impact on the evaluation of the "goodness" of routing architectures.

Detailed-router effect. As discussed earlier, the Sega detailed router has several different cost functions. Table 3 lists the average net delays in the benchmark circuits for each cost function. The Completion cost function shows that focusing only on routability does not minimize routing delays, as expected. The Seg_Len delay indicates that very poor speed performance results if the router considers only minimizing wastage of parts of wire segment. This function prevents the assignment of long wires to short connections to minimize capacitive loading; however, comparing it to the Num_Seg results shows this is a poor strategy. Minimizing the number of segments that connections pass through results in low delays. This seems to be the most important consideration because combining Num_Seg with Seg_Len worsens the results.

For Analytic_Model, Sega calculates accurate estimates of real delays. Comparing Num_Seg to Analytic_Model shows that a cost function that simply counts the number of switches traversed by a connection is a good approach. This is not to imply that other factors, such as the fanout of nets, do not affect routing delays; rather, the dominant factor is the number of switches traversed in series.

Finally, comparing the bottom row in Table 3 with the others indicates that considering multipoint nets instead of just two-point connections has a positive effect on speed performance. When the router ignores multipoint nets, it may use more wire segments and switches than the circuit actually needs in places where two-point connections on the same net overlap. This results in an increase in the net's parasitic capacitance and adds to its propagation delay.

Table 3. Effect of detailed-router cost functions on routing delays.

Sega cost function	Average net delay (ns)
Completion	12.8
Seg_Len	16.9
Num_Seg	11.9
Seg_Len + Num_Seg	13.3
Analytic_Model	11.9
Net_Routing	10.1

For Net_Routing, Sega tries to reassemble multipoint nets by focusing not only on speed performance (using Analytic_Model), but also on reusing wire segments for multiple connections that are part of the same net. Table 3 shows that Net_Routing achieves the best speed performance results. This is a valuable result because it shows the importance of efficiently routing multipoint nets. Thus, if the global router does not handle multipoint nets, the detailed router should.

Having established that the detailed-router cost function can significantly affect speed performance, it is necessary to examine the effects of Sega's cost functions on chip area. Figure 5a shows the area results of the routability-oriented Completion and speed-oriented Net_Routing cost functions. The curves of the other cost functions were between these two. To obtain these results, we set the FPGA's number of tracks per channel W to a small value (equal to the maximum channel density after global routing) and attempted detailed routing with Sega. As long as detailed routing failed, we incremented W by one until eventually Sega routed 100% of the circuit connections. We performed this for different segment lengths ranging from 1 to 8. In each case, for simplicity, all tracks had the same segment length.

From Table 3 and Figure 5a, we see that Net_Routing has the best speed performance result, but requires the most area. The Completion function achieves less than ideal routing delays, but has the best area results. An intuitive conclusion is that a good FPGA CAD routing tool should consider both speed performance and area instead of focusing on just one goal.

Channel segmentation effects. This section examines the effects on circuit implementation of the wire segment lengths in the routing channels. Stated differently, given W tracks per channel, what percentage of tracks should have

wire segments of lengths 1, 2, 3, and so on? This is the channel segmentation problem.

As a first experiment, we set all tracks in the FPGA to have the same segment length and tried different lengths from 1 to 8. We carried out two separate experiments—for area and for speed performance—and averaged the two results over all benchmark circuits (Figure 5b). For both experiments, we used Sega's Net_Routing cost function, so that the results would illustrate only the effects of wire segment length. The solid curve, which also appears in Figure 5a, shows that increasing segment length has a definite impact on the number of tracks.

The dotted curve in Figure 5b shows the effect of channel segmentation on speed performance. Segment length can have a significant impact on speed performance: Average net delay decreases by more than 40% as the segment length increases from 1 to 4. However, increasing the segment length beyond 4 slightly degrades performance. Net delays initially improve as segments become longer because the number of switches that connections pass through decreases. However, once the segments become longer than the connections themselves, the excess capacitance results in increased delays.

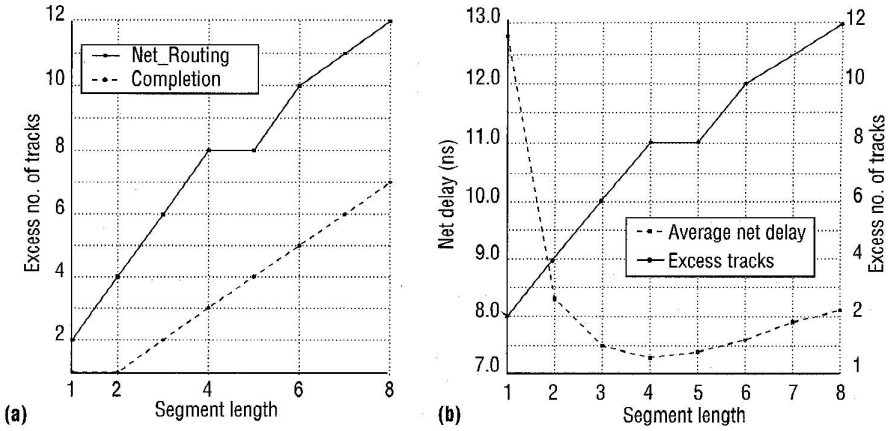


Figure 5. *Effect of: Sega's cost functions on area* ***(a)*** *and segment length on speed and area* ***(b).***

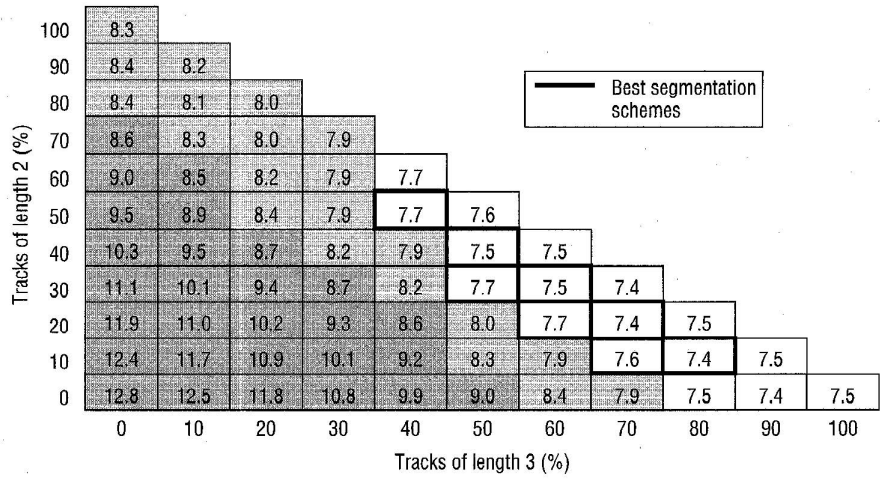


Figure 6. *Routing delays of segmentation schemes (ns).*

The increase in the number of tracks per channel (solid curve) as segment length increases not only wastes chip area, but also degrades speed performance, since extra tracks imply additional capacitive loading on logic-block pins. Considering both curves in Figure 5b, it seems likely that to decrease the interconnect delays and yet avoid an excessive number of tracks, FPGAs should use a mixture of tracks, some with long segments and others with short ones.

We repeated this experiment with channels having combinations of length-1, -2, and -3 segments; the results appear in Figure 6. Each entry represents the average net delay (in nanoseconds) produced by a particular segmentation scheme. The horizontal axis represents the percentage of tracks that are of length 3; the vertical axis, length 2; any remaining tracks are of length 1. In the figure, the shades of gray represent relative speed performance; the lighter the shade, the faster the circuit.

The minimum delay is 7.4 ns, and several segmentation schemes are close to this value. A characteristic shared by all such segmentations is that more than 80% of their tracks have segment lengths greater than 1. This leads to a fundamental conclusion that most tracks in a channel should have wire segments longer than length 1. In fact, since the result for 100% length-3 segments is near the minimum, it may be acceptable for none of the tracks to have length-1 segments. We thus pursued this last point further.

Including some tracks with length-1 and -2 segments

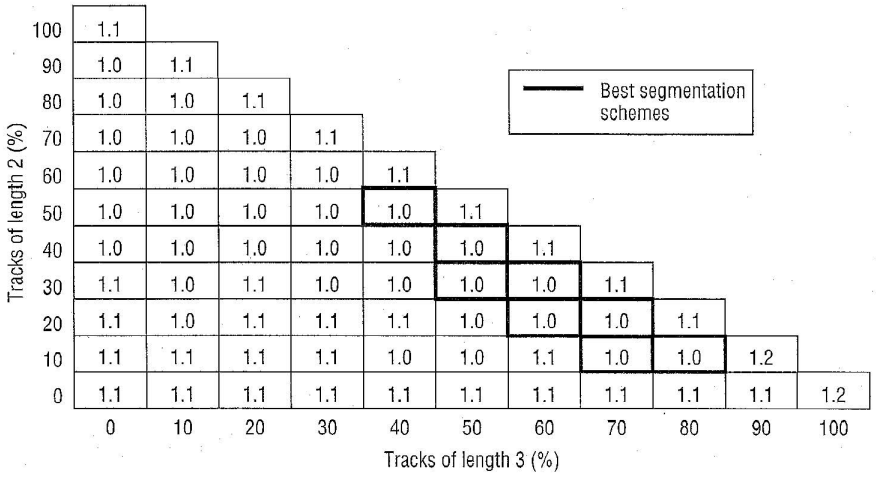


Figure 7. *Routing delays of short connections (ns).*

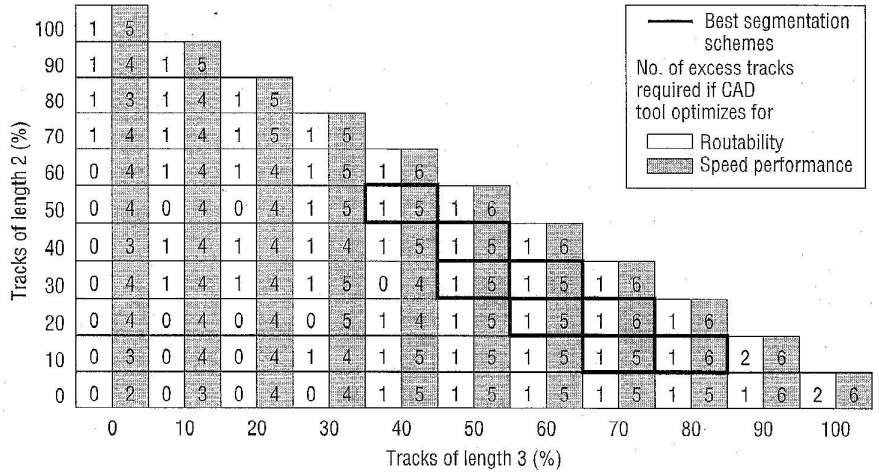


Figure 8. *Area penalty for segmentation schemes.*

should provide speed performance savings, particularly for short connections. Since Figure 6 seems to contradict this, we performed an additional experiment in which, for all Figure 6 segmentations, we measured delays only for connections that span one or two logic blocks (see Figure 7). Averaging the results in the figure shows that including length-1 segments improves the performance of short connections by about 12%. This additional information leads us to select the combinations indicated by bold outlines in Figures 6 and 7 as the best segmentation scheme candidates.

Since increasing segment length reduces net delay, we performed an additional experiment to extend segment lengths beyond the limit of 3 used in Figures 6 and 7. We repeated our earlier experiments for all possible combinations of tracks with segments of lengths 1 to 4. The results showed a small improvement (a minimum delay of 7.1 ns) over the Figure 6 values. Thus, for speed performance, it is best to include wire segments of length 3 or 4, but longer segments are not advantageous. The dotted curve in Figure 5b also supports this, indicating the minimum delay for segment lengths of around 3 or 4.

Interestingly, this result is fairly consistent regardless of the benchmark circuit size. A possible reason is that circuits tend to contain interconnected clusters of logic, rather than widely connected logic across the whole circuit. A commercial product with wire segments of lengths 1, 2, and 4 is the Lucent ORCA (Optimized Reconfigurable Cell Array).

These conclusions consider only speed performance and do not account for chip area effects. However, Figure 5b showed that increasing segment lengths can greatly affect the number of tracks, and thus we should not ignore area. To address this point, Figure 8 gives for each segmentation scheme from Figure 6 the number of tracks in excess of the maximum channel density. Consider first only the numbers in the shaded columns of Figure 8, which provide the excess tracks for the CAD tool cost function (Net_Routing) in Figure 6.

The numbers show that the best segmentation schemes from Figure 6 imply five or six extra tracks per channel. For the routed circuits, this corresponds to a 30% increase in tracks. Now consider the unshaded columns in Figure 8. They list excess tracks required if the CAD tools focus only on routability. For the best segmentation schemes, optimization for routability would require only one extra track. From this result we conclude that, although CAD tools must use speed performance cost functions to achieve low routing delays, they should focus on routability for connections with delays

that are not important. Thus, the implementation will require fewer tracks per channel. Since in real circuits many nets are not time critical, employing a combination of speed- and area-based cost functions should be feasible in practice.

THE NATURE OF AN EXPERIMENTAL STUDY such as this implies that the procedure used in some way biases the results. For instance, our CAD tools affect the results, which are also limited to the set of benchmark circuits used. For these reasons, we do not overly stress the exact values of data presented here. Rather, the appropriate viewpoint is that the absolute numbers would be different with another set of CAD tools and/or benchmark circuits, but the general conclusions should remain the same.

As stated earlier, this research most directly applies to array-based FPGA architectures using SRAM technology, like those from Xilinx. However, we believe that readers can consider much of the article in a more qualitative way that is applicable to other FPGA design styles. We have addressed some fairly general issues:

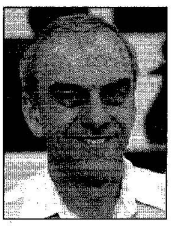
- CAD routing tools should focus on timing delays for only time-critical nets. Such tools should route nets that are not time critical with more consideration to routability. For our results, the penalty in number of tracks is about 30% if the router always focuses on timing.
- Wire-segment lengths dramatically affect speed performance. We show that for SRAM-based FPGAs this effect can be as much as 50%.
- For speed performance, the important issue is the number of switches that signals must pass through in series. Capacitive loads associated with wire segments longer than the connections that use them are insignificant in comparison to the number of switches traversed. D&T

References

1. K. Zhu and D. Wong, "On Channel Segmentation Design for Row-Based FPGAs," *Proc. Int'l Conf. Computer-Aided Design*, IEEE Computer Society Press, Los Alamitos, Calif., 1992, pp. 26-29.
2. K. Roy and M. Mehendale, "Optimization of Channel Segmentation for Channelled Architecture FPGAs," *Proc. 1992 Custom Integrated Circuits Conf.*, IEEE, Piscataway, N.J., 1992, pp. 4.4.1-4.4.4.
3. H. Hsieh et al., "A Second Generation User-Programmable Gate Array," *Proc. 1987 Custom Integrated Circuits Conf.*, IEEE, 1987, pp. 515-521.
4. H. Hsieh et al., "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays," *Proc. 1990 Custom Integrated Circuits Conf.*, IEEE, 1990, pp. 31.2.1-31.2.7.
5. S. Brown, G. Lemieux, and M. Khellah, "Segmented Routing for Speed-Performance and Routability in Field-Programmable Gate Arrays," *J. of VLSI Design*, Vol. 4, No. 4, 1996, pp. 275-291.
6. L.E. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," *Proc. 1995 Int'l Symp. on FPGAs*, Assoc. for Computing Machinery, New York, 1995, pp. 111-117.
7. Y.-S. Lee and A.C.-H. Wu, "A Performance and Routability Driven Router for FPGAs Considering Path Delays," *Proc. 1995 Design Automation Conf.*, ACM, 1995, pp. 557-561.
8. Y.-L. Wu and M. Marek-Sadowska, "Orthogonal Greedy Coupling—A New Optimization Approach to 2-D FPGA Routing," *Proc. 1995 Design Automation Conf.*, ACM, 1995, pp. 568-573.
9. M.J. Alexander and G. Robins, "New Performance-Driven FPGA Routing Algorithms," *Proc. 1995 Design Automation Conf.*, ACM, 1995, pp. 562-567.
10. M. Khellah, "Minimizing Interconnection Delays in Array-Based FPGAs," master's thesis, Univ. of Toronto, Dept. of Electrical and Computer Eng., Jan. 1994.
11. J. Rubinstein, P. Penfield, and M. Horowitz, "Signal Delay in RC Tree Networks," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-2, No. 3, Jul. 1983.

Stephen Brown's biography appears on p. 15.

Muhammad Khellah is currently pursuing a PhD at the University of Waterloo, Canada, where he is working on CAD for low-power design. Khellah received the BS degree in computer engineering from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, and the MASc degree in electrical and computer engineering from the University of Toronto. He is a student member of the IEEE.



Zvonko Vranesic is a professor in the Department of Electrical and Computer Engineering of the University of Toronto. His research interests include computer architecture, VLSI systems, local area networks, and many-valued switching systems. Vranesic received the BASc, MASc, and PhD degrees from the University of Toronto. He is the coauthor of three books and served as the chair of the 18th International Symposium on Computer Architecture.

Address questions or comments about this article to Stephen Brown, Department of Electrical and Computer Engineering, University of Toronto, 10 Kings College Road, Toronto, Ontario, Canada, M5S 3G4; brown@eecg.toronto.edu.