A Crosstalk-Aware Timing-Driven Router for FPGAs

Steven J. E. Wilton
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, B.C., Canada
stevew@ece.ubc.ca

ABSTRACT

As integrated circuits are migrated to more advanced technologies, it has become clear that crosstalk is an important physical phenomenon that must be taken into account. Crosstalk has primarily been a concern for ASICs, multi-chip modules, and custom chips, however, it will soon become a concern in FPGAs. In this paper, we describe the first published crosstalk-aware router that targets FPGAs. We show that, in a representative FPGA architecture implemented in a 0.18µm technology, the average routing delay in the presence of crosstalk can be reduced by 7.1% compared to a router with no knowledge of crosstalk. About half of this improvement is due to a tighter delay estimator, and half is due to an improved routing algorithm.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids – *placement and routing.*

General Terms

Algorithms.

Keywords

Field-Programmable Gate Arrays, Routing Algorithms, Crosstalk.

1. INTRODUCTION

As integrated circuits are migrated to more advanced technologies, new physical phenomena have come to light. Recently, crosstalk, or inter-wire capacitance, has become a critical concern among integrated circuit designers. Crosstalk occurs when a change in voltage on one trace causes a change in voltage on a nearby trace. This can cause a system to fail in two ways:

1. If the affected trace (called the victim) is a dynamic node, a clock signal, or an asynchronous set or reset, the induced pulse might cause the circuit to change state incorrectly. This sort of functional error is also a concern in mixed-signal designs, in which the victim trace carries an analog signal.
2. The switching time of the victim may be affected. If the switching time increases too much, the path delay may increase. This may cause timing failures.

As the minimum feature size of integrated circuits decreases, crosstalk becomes more important. In part, this is because metal traces are getting taller and narrower; by 2004, it is expected that a trace will be three times taller than it is wide [1]. This increases the effective capacitance between neighbouring traces, and hence increases the potential that crosstalk will cause system failure.

To ensure crosstalk does not cause system failure, it must be taken into account by CAD tools. There has been much work in routers that understand crosstalk [2,3,4,5,6,7,8,9]. There are two types of crosstalk-aware routers:

1. Routers that strive to ensure the correctness of a circuit: Given a list of crosstalk constraints, these routers attempt to find a minimum-cost routing that meets the given crosstalk constraints. The constraints are usually given to ensure specific nets (such as clock nets) are not routed adjacent to more than a specified number of other nets.
2. Routers that optimize for delay in the presence of crosstalk: The effects of crosstalk can be modeled by an increase or decrease in the wiring capacitance of the victim net. This modified capacitance can then be used in timing models.

In this paper, we focus on the second goal.

All of the previously published crosstalk-aware routers target standard cell, multi-chip modules, or full custom designs. In these technologies, inter-wire capacitance is the dominant contribution to the capacitance of a wire. Thus, crosstalk-induced changes in this capacitance will have a significant effect on the delay of a net. In an FPGA, however, the inter-wire capacitance is comparable to the capacitances of the buffers and switches attached to the trace, so crosstalk will have a smaller effect on the speed of an FPGA circuit. Yet, developing crosstalk-aware routing algorithms for FPGAs is still important. As we will show in this paper, even at 0.18µm, significant performance improvements can be obtained by taking crosstalk into account. Technology shrinks will continue, and eventually crosstalk will be a dominating factor in FPGA performance. When this happens, algorithms such as the one described in this paper will become essential.

Simply extending the ASIC-based algorithms to FPGAs will not work well. The primary reason is that a standard cell (or similar) router has much more flexibility than an FPGA router. One of the primary goals of a crosstalk-aware router is to route critical signals away from other nets, so that the critical signals are less likely to be crosstalk victims. In an ASIC router, there are far more options for each net. Consider Figure 1. In this diagram, a critical net connecting to the logic block pin is to be routed. Only tracks B and D can be used, since these are the only two tracks which can be connected to the logic block pin. Both B and D are bad choices; both are adjacent to a previously-routed net. A potential solution is to re-route the pre-routed net; however, this net will have its own constraints regarding which tracks can be used. In an ASIC router, the task is easier; in most cases, any available horizontal position (A to E) could be used. In addition, in an ASIC router, if a previously-routed net does need to be moved, it can likely be done easily.

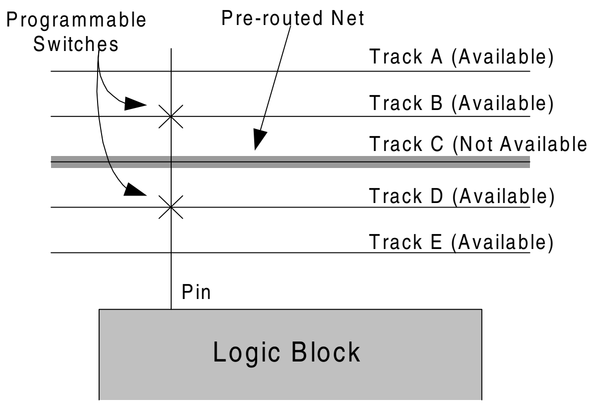


Figure 1: An Example Routing on an FPGA

Because of this lack of flexibility, it is important that crosstalk be handled not just as a post-processing step, as it is in [5,9], or during detailed routing, as it is in [2,3,6,7,8]. If an FPGA router can only optimize for crosstalk during detailed routing, the limited switch and connection block flexibilities make it unlikely that any optimizations will be possible. Thus, it is important to take crosstalk into account during the entire routing process.

In this paper, we present a combined global/detailed FPGA router which optimizes for delay in the presence of crosstalk. The algorithm is based on the VPR router [10] which is representative of industrial tools. The algorithm applies regardless of the underlying architecture and technology; we will evaluate it using an island-style FPGA implemented in a 0.18µm CMOS process.

This paper is organized as follows. Section 2 reviews the baseline VPR routing algorithm. Section 3 then shows how this algorithm can be extended to understand crosstalk. Finally, Section 4 presents experimental evidence that the algorithm works well across a wide variety of architectures.

2. BASELINE ALGORITHM

Our algorithm is based on the VPR router described in [10], which uses an algorithm similar to that employed by Pathfinder, described in [11]. Nets are routed sequentially using a maze-routing algorithm. Initially, nets are allowed to share physical tracks. Once all nets have been routed, the cost of two nets sharing a track is increased slightly. Each net is then ripped-up and re-routed. This is repeated for several iterations; each time the cost of sharing becomes slightly higher. When, at the end of an iteration, no track is shared between more than one net, a legal routing has been found, and the algorithm terminates.

During maze-routing, the fitness of each potential segment *n* that might be added to the net is evaluated using the following cost function:

$$Cost(n) = Crit \bullet delay(n) + (1-Crit) \bullet b(n) \bullet h(n) \bullet p(n) \qquad (1)$$

where *delay(n)* is the Elmore delay of the segment *n*, and *b(n)*, *h(n),* and *p(n)* are the base cost, the historical congestion cost, and the present congestion cost of using segment *n*. The quantity *Crit* is called the criticality of the currently routed net (or the current sink on the currently routed net if the net has more than one sink). The criticality of a given net is close to 1 if the net is close to being on the critical path of the circuit. Nets that are not on the critical path have a lower criticality (an equation for the criticality of a net is given in [10]).

Notice that Equation 1 contains two terms. The first represents the delay of the currently routed net, while the second represents the congestion cost for the current segment. Nets with a high value of *Crit* (ie. nets on or near the critical path) are thus routed primarily for speed, while other nets are routed primarily for congestion. This ensures that, as routing progesses, nets which are not critical are moved away from congested regions.

The delay term in Equation 1 is computed using the VPR timing model. The timing model uses the Elmore delay, along with a detailed description of the underlying architecture (including segmentation strategy, buffer placement and size, parasitic resistances and capacitances, etc). In [10], it is shown that the Elmore delay gives good fidelity; that is, it correctly ranks choices during routing.

3. ENHANCED ALGORITHM

In this section, we describe how the VPR router can be enhanced to optimize for delay in the presence of crosstalk. There are components within VPR that are of concern: the timing model, which is used to evaluate potential routes (as well as the final solution), and the cost-function used by the maze router. Both are treated separately below.

3.1 Timing Model

There has been a significant amount of work developing models that describe the effects of crosstalk [3,12,13,14,15]. In this paper, we assume a simple model, in which crosstalk between two traces is modeled by a change in the effective capacitance seen by both traces. The magnitude of this change depends on the relative switching activity of the two traces.

The capacitance of a trace *i* can be written as:

$$C_i = C_{fixed} + \sum_{j \,\varepsilon\, T,\, j \neq i} \frac{\lambda_{i,j} C_c \, l_{i,j}{}^{\eta}}{d_{i,j}{}^{\gamma}} \qquad (2)$$

where C_{fixed} is the capacitance of the metal trace itself, as well as any switches or buffers attached to the trace, *T* is the set of all traces, $l_{i,j}$ is the distance that traces *i* and *j* are adjacent, $d_{i,j}$ is the distance between trace *i* and *j*, $\lambda_{i,j}$ is a parameter described below, and γ, η, and C_c are process-dependent constants (typical values of γ and η are 1.8 and 0.92 respectively). [4].

Note that Equation 2 assumes traces *i* and *j* have no trace between them. In an FPGA, this is only true for neighbouring tracks within a channel. Thus, the summation only needs to be carried out for those wires. By assuming all tracks are within a channel are separated by the same distance, by combining $d_{i,j}$ with the technology dependent terms, and approximating η as 1, we can write:

$$C_i = C_{fixed} + \sum_{j \,\varepsilon\, N(i)} \lambda_{i,j} C_{couple} \, l_{i,j} \qquad (3)$$

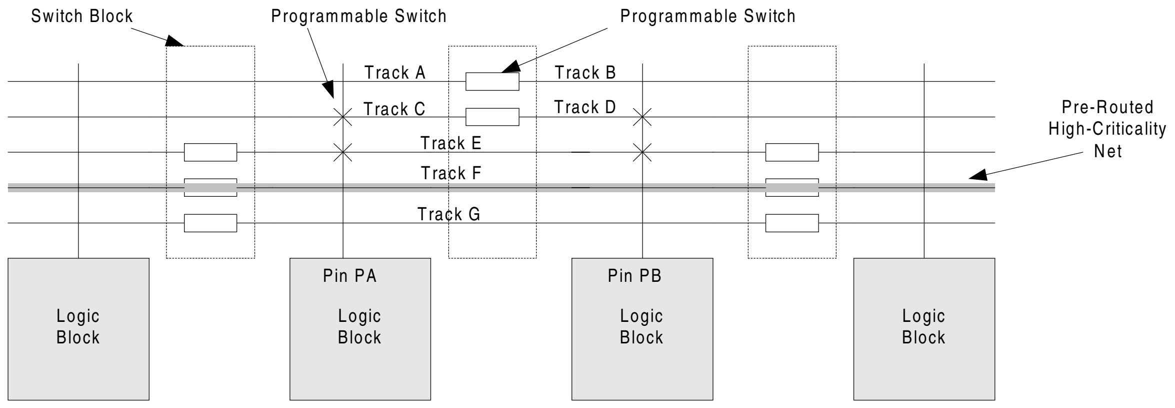


Figure 2: Example Illustrating Crosstalk Routing

where *N(i)* is the set of neighbours of *i* (note that a track may have more than two neighbours if the track spans more than one logic block). From Equation 3, it is clear that the capacitance of a track depends on the capacitances of the neighbouring track(s) within the same channel.

The parameter $\lambda_{i,j}$ takes into account the switching activity between tracks *i* and *j*. If the two tracks are switching in the same direction, $\lambda_{i,j}$ can be approximated as 0, since there is no induced crosstalk for signals switching together. If the two signals are switching in the opposite direction, $\lambda_{i,j}$ can be approximated as 2. If only one of the signals is changing, $\lambda_{i,j}$ can be approximated as 1. During routing, it is difficult to determine the relative switching activities of each signal. Thus, our timing model makes the following pessimistic assumptions when computing the effective capacitance of track *i*:

1. For all neighbours *j* which are used to carry a signal, we assume the worst-case crosstalk ensues (ie. $\lambda_{i,j}$ is 2).
2. For all neighbours *j* which are unused, we assume that *j* is tied to a constant voltage, so $\lambda_{i,j}$ is 1.

Thus, for any given routing, the effective capacitance of each track can be computed, and this can be used in the Elmore delay to estimate the delay of the net.

3.1 Cost Function

The intelligence of any iterative maze-router is encapsulated in its cost function. This subsection describes how the VPR cost function can be modified to account for crosstalk.

There are two ways in which the new cost function takes crosstalk into account. The first is that the *delay(n)* quantity from Equation 1 is modified to use the new timing model from Section 3.1. In this way, the delay of using a segment is computed using not only the capacitance of the segment itself, but the extra capacitance due to any previously-routed nets. This will tend to route nets away from other nets, thereby lowering the crosstalk effect, and possibly lowering the critical path delay of the circuit.

This is not enough, however. Consider Figure 2. Suppose a high-criticality net (net on the critical path) has previously been routed onto track F. Now suppose logic block pins PA and PB are to be connected using a low-criticality net (a net that is not on the critical path). There are two options. The pins could be connected using track E which spans two logic blocks. Alternatively, the pins could be connected using tracks C and D (there is a programmable switch between tracks C and D). In a router which does not understand crosstalk, the first option (track E) would be preferred, since only one segment needs to be used.

Now consider a crosstalk-aware router, in which the *delay(n)* quantity from Equation 1 is modified to use the new timing model as described above. Depending on the criticality of the net being routed, either the first option (track E) or second option (tracks C and D) may be chosen. The second option will suffer less crosstalk, but recall from Equation 1, the net delay (and hence the crosstalk penalty) is multiplied by the criticality of the net being routed. In our example in Figure 2, we are routing a low-criticality net, meaning the crosstalk contribution to the overall cost function will be small. Thus, the first option (Track E) may be preferred. However, this is probably a bad choice, since using track E will also increase the crosstalk seen by the critical path net routed on track F. This will cause the critical path of the system to become longer. This motivates our need for a mechanism for low-priority nets to understand the anti-social behaviour of routing next to a high-priority net.

Such a mechanism can be implemented by modifying the cost function of Equation 1 as follows:

$$\begin{aligned} Cost(n) = {} & Crit \cdot delay(n) + penalty(n) + \\ & (1\text{-}Crit) \cdot b(n) \cdot h(n) \cdot p(n) \end{aligned} \tag{4}$$

where

$$penalty(n) = \sum_{m \in N(n)} \sum_{k \in U(m)} Crit(k)\ \Delta delay(k,m,n) \tag{5}$$

In Equation 5, *N(n)* is the set of neighbours of track *n*, *U(m)* is the set of previously-routed connections using track *m, Crit(k)* is the criticality of connection *k*, and *Δdelay(k,m,n)* is the increase in the delay of connection *k* (routed on track *m*) that occurs if a new signal is routed using track *n*. This latter quantity can be computed as:

$$\Delta delay(k,m,n) = R_{upstream,m,k}\, C_{couple}\, l_{m,n}$$

where $l_{m,n}$ is the distance that traces *m* and *n* are adjacent, C_{couple} is a technology constant as in Equation 3, and $R_{upstream,m,k}$ is the resistance between track *m* and the source of net *k*. This upstream resistance is already computed in VPR for use in the timing analyzer; thus, Equations 4 through 6 can be easily computed. Note that Equation 6 describes the increase in a delay in a two-pin net; experiments have shown that this gives good results for nets with more than two pins as well.

Circuit	Number 4-LUTs / flip-flops	Number Tracks per channel	Baseline VPR (worstcase crosstalk)		Baseline Router, Enhanced Timing		Enhanced Router and Timing Model	
			Routing Delay	Critical Path	Routing Delay	Critical Path	Routing Delay	Critical Path
apex1	696 / 0	49	7.78 ns	12.98 ns	7.60 ns	12.80 ns	7.38 ns	11.88 ns
apex3	867 / 0	49	8.94 ns	13.44 ns	8.58 ns	13.08 ns	8.46 ns	12.95 ns
apex4	1262 / 0	53	10.27 ns	14.77 ns	9.96 ns	14.46 ns	9.53 ns	14.03 ns
cordic	466 / 0	36	8.54 ns	15.15 ns	8.26 ns	14.87 ns	7.89 ns	14.50 ns
cps	749 / 0	42	9.08 ns	12.88 ns	8.76 ns	12.56 ns	7.39 ns	11.18 ns
dalu	500 / 0	33	7.92 ns	12.41 ns	7.39 ns	11.89 ns	6.93 ns	10.72 ns
dsip	1370 / 224	38	5.37 ns	7.61 ns	5.25 ns	7.49 ns	4.99 ns	7.23 ns
ex5p	1064 / 0	57	9.23 ns	14.43 ns	8.91 ns	14.11 ns	8.95 ns	14.15 ns
k2	515 / 0	40	8.49 ns	12.99 ns	8.21 ns	12.70 ns	7.13 ns	12.33 ns
pair	641 / 0	40	6.64 ns	11.14 ns	5.74 ns	10.94 ns	6.53 ns	11.73 ns
planet	266 / 6	21	4.24 ns	7.33 ns	4.12 ns	7.21 ns	4.32 ns	7.42 ns
s298	1930 / 8	38	16.97 ns	27.64 ns	16.71 ns	27.36 ns	16.74 ns	27.41 ns
s1488	296 / 6	21	4.34 ns	7.44 ns	4.12 ns	7.21 ns	3.95 ns	7.04 ns
s5378	572 / 160	38	5.68 ns	10.18 ns	5.46 ns	9.96 ns	4.90 ns	9.40 ns
sbc	372 / 27	29	5.07 ns	8.16 ns	4.93 ns	8.03 ns	4.76 ns	7.85 ns
table5	483 / 0	44	7.50 ns	12.00 ns	7.21 ns	11.71 ns	7.10 ns	11.60 ns
tseng	1046 / 385	36	8.51 ns	17.78 ns	8.35 ns	17.61 ns	8.21 ns	17.47 ns
Average			7.92 ns	12.84 ns	7.62 ns	12.59 ns	7.36 ns	12.29 ns

Table 1: Timing results assuming segment length=8, f_c=0.6, 30% extra tracks

Intuitively, the penalty term from Equation 5 represents the increase in delay of all adjacent nets weighted by the criticality of the adjacent nets. In the example of Figure 2, the new net would see a high penalty for using track E, since criticality of the adjacent net is high. Thus, the router would prefer to use tracks C and D, which, overall, should lead to a faster circuit.

4. EXPERIMENTAL RESULTS

To evaluate the proposed enhancements, we experimentally mapped seventeen large benchmark circuits onto a model FPGA. We assumed an island-style FPGA, where each logic block contains four 4-input lookup tables and four flip-flops. The switch block from [16] was used, but the number, length, and driving capacity of each routing track was varied throughout the experiments. The number of connections between each logic block pin and the adjacent routing channel (F_c in [17]) was also varied throughout the experiments. In every case, we assumed a 0.18μm CMOS process available from TSMC. Although we do not expect crosstalk to become an over-riding concern for FPGAs until they are well below 0.15μm, this section will show that, even at 0.18μm, significant performance improvements can be gained by taking crosstalk into account during routing.

Each circuit was first mapped to 4-input lookup tables and flip-flops using Flowmap/Flowpack [18]. The lookup tables and flip-flops were then packed into logic blocks using a timing-driven packing algorithm [10]. The logic blocks were then placed on an appropriately sized FPGA using the timing driven placement tool described in [19]. We then ran the baseline VPR timing-driven router to find the minimum number of routing tracks needed for 100% routability. This number was then increased by 30%, and the routing repeated, this time with both the baseline VPR algorithm and the enhanced algorithm described in Section 3.0. This "low-stress" routing is representative of the routing performed in real industrial designs. Note that , when comparing the baseline and enhanced routing algorithms, the same number of tracks is used in each case, therefore, the area required in both cases is the same.

Table 1 shows the results assuming each routing segment spans eight logic blocks, assuming the segments are separated by re-powering buffers, and assuming each logic block pin can connect to 60% of the tracks in the adjacent channel (ie. F_c=0.6). The third column shows the number of tracks used to route each circuit (this is the minimum number of tracks for each circuit increased by 30%).

When comparing the original VPR router and our enhanced crosstalk-aware router, there are two components that should be measured separately. The first component is due to the enhanced timing analyzer. The original VPR timing analyzer has no notion of crosstalk, so (to be pessimistic) we need to assume the worst-case coupling capacitance between each pair of neighbouring tracks, regardless of whether these tracks are used. Our new timing analyzer provides a tighter estimate on the speed of the circuit, by taking into account whether neighbouring tracks of a given track are used or unused. To get an idea of the amount of improvement due to the enhanced timing analyzer, we first ran the original VPR tool, and gathered the results in columns four and five of Table 1. Column five shows the critical path of each circuit, while column four shows the portion of the critical path due to routing (the routing delay). We then *used the same routing solutions* for each circuit, but used the new timing analyzer to get a tighter bound on the speed of each implementation. Columns six and seven show this data. Comparing columns four/five with six/seven, we can see that an improvement of 3.8% in the routing delay (1.9% in the critical path) was obtained by the new timing analyzer.

The second component of the improvement is due to the routing algorithm itself. The above comparisons both assumed a router that did not optimize for crosstalk. We ran our enhanced algorithm on each circuit, and gathered the results in columns eight and nine. Comparing columns six/seven to columns eight/nine, it can be seen that the new router reduces the routing delay by an additional 3.4%,

Circuit	Isolation Factor		Circuit	Isolation Factor	
	Baseline Router	Enhanced Router		Baseline Router	Enhanced Router
apex1	18.5	27.5	pair	29.3	48.3
apex3	27.0	26.9	planet	21.1	27.4
apex4	31.1	26.9	s298	26.9	32.9
cordic	19.1	33.8	s1488	16.9	26.5
cps	28.8	38.6	S5378	27.8	48.5
dalu	26.9	35.6	sbc	33.1	28.7
dsip	26.0	43.0	table5	17.8	34.2
ex5p	29.5	21.1	tseng	26.9	41.3
k2	29.8	31.1	Average	25.7	33.7

Table 2: Isolation factor results assuming segment length=8, f_c=0.6, 30% extra tracks

and the critical path by an additional 2.4%. Overall, our average routing delay is 7.1% smaller than the original VPR results, while the average critical path is 4.5% smaller.

To gain further insight into how well our routing algorithm is performing, we define the *isolation factor* of a routing as follows. For a given routing, assume *G* is the set of segments that lie on the critical path (or paths). For each element $g \varepsilon G$, define *N(g)* as the set of tracks adjacent to *g*, and define $l_{n,g}$ as the length of the adjacency between tracks *g* and *n*, in terms of logic blocks. Further define *H(n)* to be 0 if track *n* is used, and 1 otherwise. Then, the isolation factor of a routing is defined as:

$$\text{Isolation Factor} = 100 \times \frac{\sum_{g \varepsilon G} \sum_{n \varepsilon N(g)} l_{n,g} H(n)}{\sum_{g \varepsilon G} \sum_{n \varepsilon N(g)} l_{n,g}} \qquad (7)$$

Informally, the isolation factor is an indication of how well the router is isolating the critical path from other nets. A value of 0 indicates every segment of the critical path(s) is adjacent to another net, while a value of 100 means that every segment of the critical path(s) is not adjacent to any other net. Table 2 shows the isolation factors for both the base and enhanced routers for each of our benchmark circuits. As the table shows, the isolation factor is increased by, on average, 31% in the enhanced router.

4.1 Parameter Sweep: Extra Tracks per Channel

To investigate the effectiveness of the algorithm across different architectures, we swept three parameters: the number of extra tracks per channel, the number of connections per logic block pin (F_c), and the length of each segment.

First consider the extra number of tracks in each channel. The results in Tables 1 and 2 assume each channel has 30% more tracks the minimum number of tracks required for 100% routability. To investigate the effect that the number of tracks has on our algorithm, we varied the number of extra tracks, and in each case, measured the isolation factor and the routing delay. Intuitively, we would expect that as the number of extra tracks goes up, the new algorithm is better able to reduce crosstalk. Figure 3 shows that this is true. The first graph in Figure 3 shows the isolation factor (averaged over all benchmark circuits) of the enhanced algorithm and the baseline algorithm (with the enhanced timing model) as a function of the number of extra tracks. The second graph in Figure 3 shows the percent improvement in the isolation factor. The trend is clear: the improvement increases until the isolation factor of the enhanced algorithm approaches 100%. At this point, there is no further improvement possible in the enhanced algorithm isolation factor, so the percentage improvement drops. Figure 4 shows the affect on the routing delay of the two algorithms (again, the baseline algorithm includes the enhanced timing model). The trends in these graphs are less clear; but it appears that the improvement in routing delay improves as the number of extra tracks increases.

4.2 Parameter Sweep: Connection Block Flexibility

Now consider the number of connections per logic block pin (F_c). Intuitively, the larger the value of F_c, the more flexibility that is available to the enhanced router. Therefore, we would expect that as F_c increases, the new algorithm is better able to isolate tracks. On the other hand, a larger F_c means there is more fixed capacitance attached to each routing track; this capacitance will tend to reduce the improvement obtained by isolating the critical path. Figures 5 and 6 show experimental results. Clearly, both the isolation factor improvement and the routing delay improvement increase as F_c increases.

4.3 Parameter Sweep: Segment Length

Finally, we investigated the effect of the length of each routing segment on the effectiveness of our algorithm. Figures 7 and 8 show the isolation factor experimental results and the routing delay experimental results as a function of segment length. In each case, an architecture consisting of buffered switches is assumed; the size of each buffer increases as the segment length increases (for each segment length, the optimum buffer size was found). As the graphs show, the improvement in isolation factor increases as the segment length increases, but no trend is observable in the delay graph. We experimented with other architectures, and got similar results.

5. CONCLUSIONS

In this paper, we have presented an FPGA router than optimizes for delay in the presence of crosstalk. By optimizing an existing FPGA router, we were able to improve the average routing delay of circuits by 7.1% compared to an FPGA router which assumes the worst-case coupling capacitance between all neighbouring tracks. About half of this improvement was due to a tighter delay estimator, and half was due to the new routing algorithm.

There are many ways to improve the results in this paper. One way is to relax the pessimistic assumption that two neighbouring tracks always result in the worst-case coupling capacitance. One way to do this would be to assign "change windows" or "bins" for each segment; a bin is a subset of the clock period in which a given signal might change. Neighbouring tracks which carry signals with non-overlapping bins will not result in the worst-case crosstalk; taking advantage of this would give additional freedom to the router. Of course, the effectiveness of this will depend on how tight the bin boundaries can be (if, for most signals, the bin spans the entire clock period, clearly no optimizations beyond those in this paper would be possible). In addition, more advanced crosstalk models which are currently being developed by researchers would give more accurate crosstalk estimates, and possibly better overall results. As technology shrinks, and crosstalk becomes more and more significant, further optimizations such as these will become more and more important.

6. ACKNOWLEDGMENTS

Funding was provided by the British Columbia Advanced Systems Institute and the Natural Sciences and Engineering Research Council of Canada. The author wishes to thank the Canadian Microelectronics Corporation for providing process data and Mike Sheng and Elias Ahmed for encapsulating this data into a VPR-readable format.

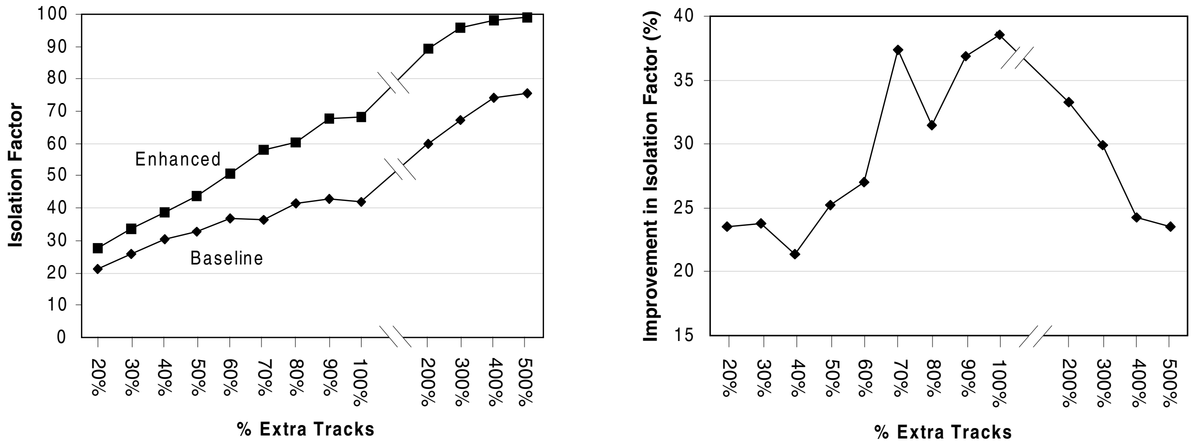


Figure 3: Effect of Number of Extra Tracks on Isolation Factor

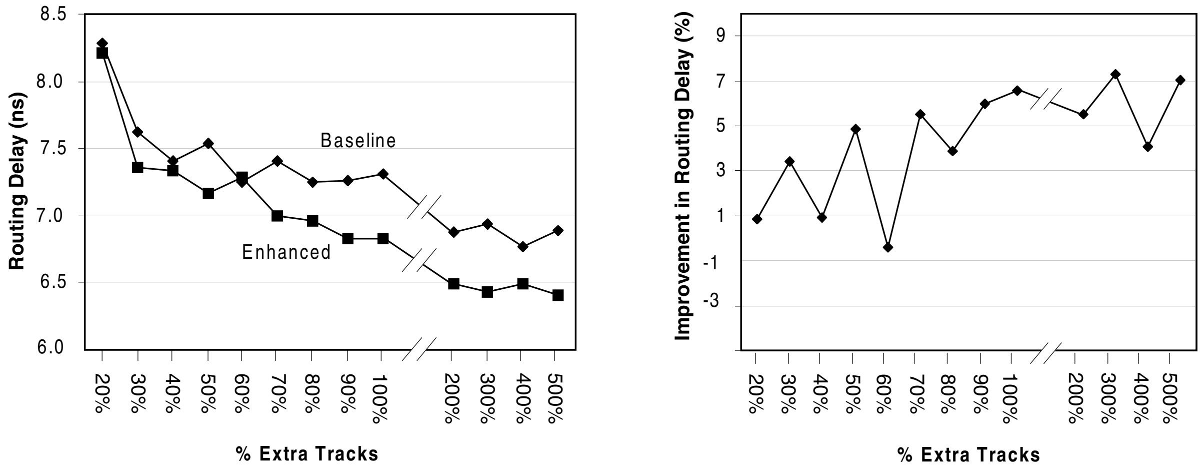


Figure 4: Effect of Number of Extra Tracks on Routing Delay

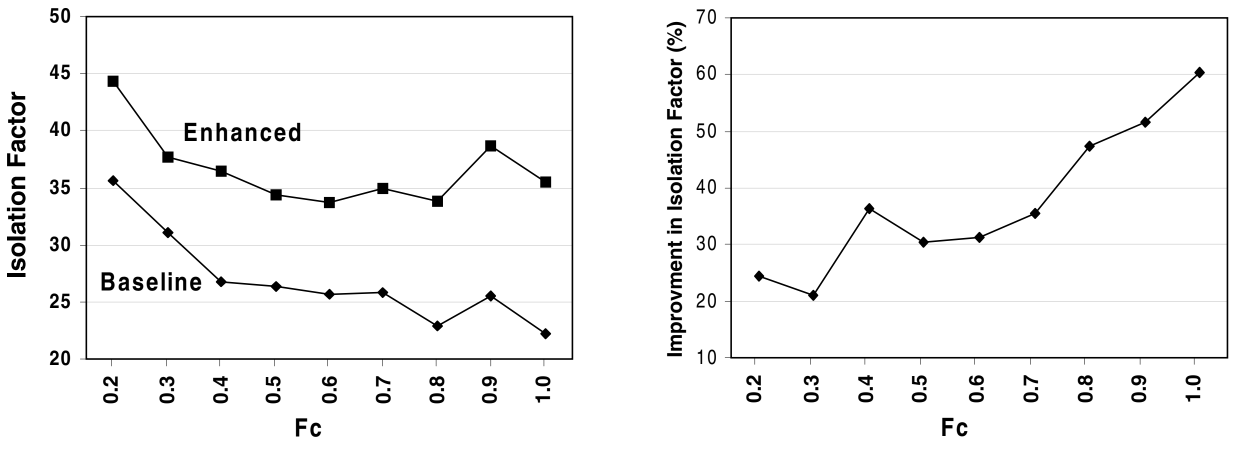


Figure 5: Effect of F_c on Isolation Factor

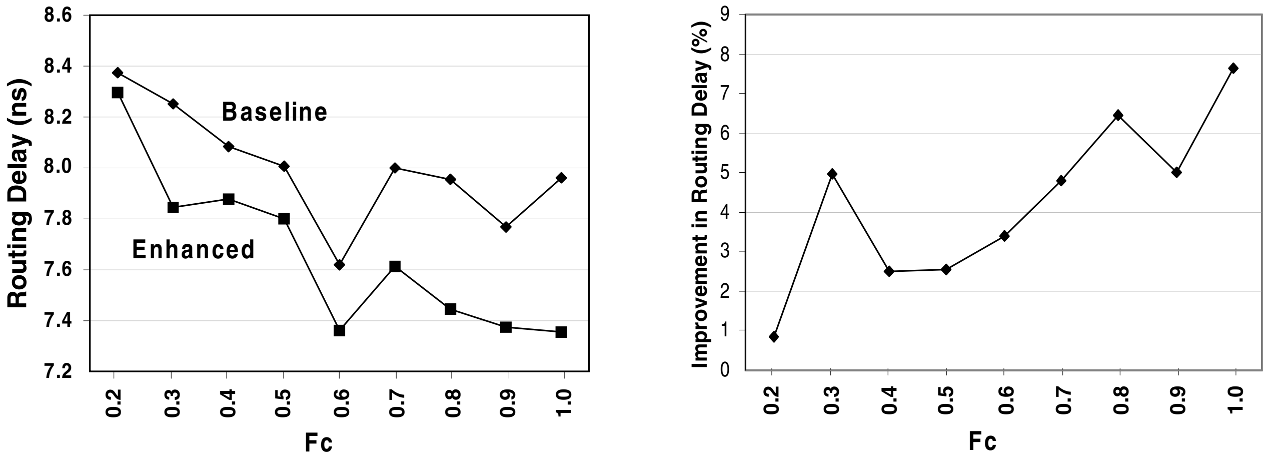


Figure 6: Effect of F_c on Routing Delay

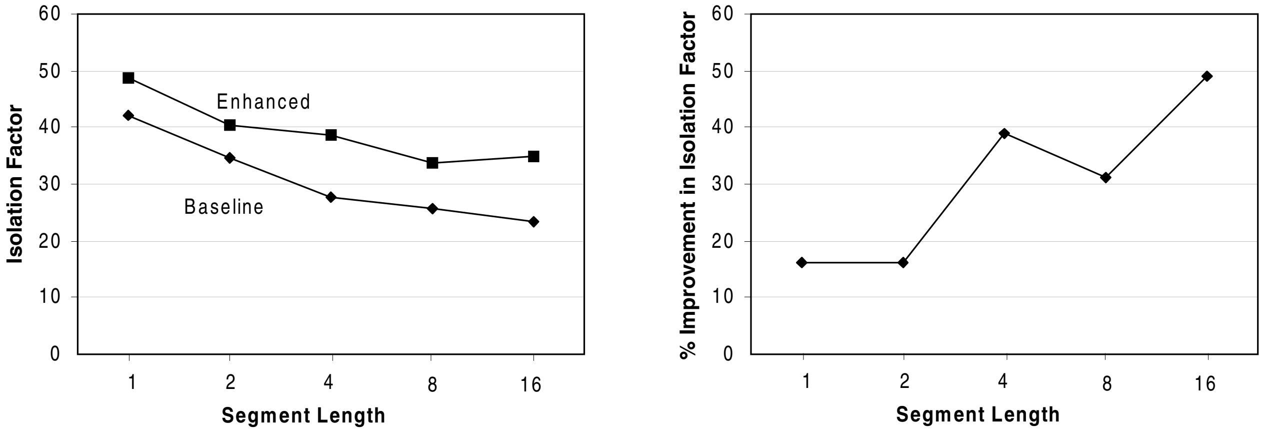
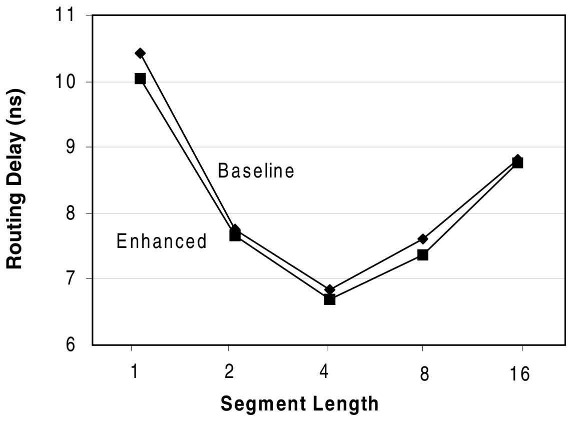


Figure 7: Effect of Segment Length on Isolation Factor



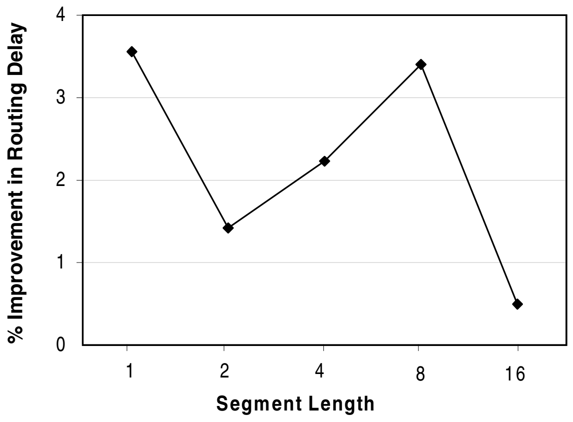


Figure 8: Effect of Segment Length on Routing Delay

REFERENCES

[1] The National Technology Roadmap for Semiconductors, Semiconductor Industry Association, 1994.

[2] U. Choudhury, A. Sangiovanni-Vincentelli, "Constraint-Based Channel Routing for Analog and Mixed Analog/Digital Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 4, April 1993, pages 497-510.

[3] A. Vittal, M. Marek-Sadowska, "Crosstalk Reduction for VLSI," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 3, March 1997, pages 290-298.

[4] P. Saxena, C.L. Liu, "A Postprocessing Algorithm for Crosstalk-Driven Wire Preturbation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 6, June 2000, pages 691-702.

[5] T. Gao, C.L. Liu, "Minimum Crosstalk Channel Routing," *IEEE Transactions on Computer-Aided Design*, Vol. 15, No. 5, May 1996, pages 465-474.

[6] T. Miyoshi, S. Wakabayashi, T. Koide, N. Yoshida, "An MCM Routing Algorithm Considering Crosstalk," in *Proceedings of the International Symposium on Circuits and Systems*, May 1995, volume 1, pages 211-214.

[7] H. Zhou, D.F. Wong, "An Optimal Algorithm for River Routing with Crosstalk Constraints," in *Proceedings of the International Conference on Computer-Aided Design*, November 1996, pages 310-315.

[8] D. Wang, E. Kuh, "A Performance-Driven MCM Router with Special Consideration of Crosstalk Reduction," in *Proceedings of Design Automation and Test in Europe*, 1998, pages 466-470.

[9] K. Jhang, S. Ha, C.S. Jhon, "COP: A Crosstalk Optimizer for Gridded Channel Routing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 4, April 1996, pages 424-429.

[10] V. Betz, J. Rose, A. Marquardt, "*Architecture and CAD for Deep-Submicron FPGAs*," Kluwer Academic Pub., 1999.

[11] C. Ebeling, L. McMurchie, S.A. Hauck, S. Burns, "Placement and Routing Tools for the Tryptic FPGA," IEEE Transactions on VLSI Systems, Vol. 3, No. 4, December 1995, pages 473-482.

[12] T. Xiao, M. Marek-Sadowska, "Efficient Delay Calculation in the Presence of Crosstalk", in *Proceedings of the International Symposium on Quality of Electronic Design*, March 2000.

[13] B. Franzini, C. Forzan, D. Pandini, P. Scandolara, A. Dal Fabbro, "Crosstalk Aware Static Timing Analysis: A Two Step Approach," *in Proceedings of the International Symposium on Quality of Electronic Design*, March 2000.

[14] P. Tehrani, S. Chyou, U. Ekambaram, "Deep Sub-Micron Static Analysis in Presence of Crosstalk,"in *Proceedings of the International Symposium on Quality of Electronic Design*, March 2000.

[15] M. Becer, I. Hajj, "An Analytical Model for Delay and Crosstalk Estimation with Application to Decoupling," in *Proceedings of the International Symposium on Quality of Electronic Design*, March 2000.

[16] M.I. Masud, S.J.E. Wilton, "A New Switch Block for Segmented FPGAs," *in Lecture Notes in Computer Science 1673*, Springer-Verlag, pages 274-281

[17] J.Rose, S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," *IEEE JSSC*, Vol. 26, No. 3, March 1991, pages 277-282.

[18] J. Cong, Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 1, January 1994, pages 1-12.

[19] A. Marquardt, V. Betz, and J. Rose, "Timing-Driven Placement for FPGAs," *in Proceedings of the International Symposium on Field-Programmable Gate Arrays*, February 2000, pages 203-21