Mixing Buffers and Pass Transistors in FPGA Routing Architectures

Mike Sheng and Jonathan Rose
Department of Electrical and Computer Engineering, University of Toronto
Toronto, Ontario, Canada M5S 3G4
{sheng, jayar}@eecg.toronto.edu

Abstract

The routing architecture of an FPGA consists of the length of the wires, the type of switch used to connect wires (buffered, unbuffered, fast or slow) and the topology of the interconnection of the switches and wires. FPGA Routing architecture has a major influence on the logic density and speed of FPGA devices. Previous work [1] based on a 0.35um CMOS process has suggested that an architecture consisting of length 4 wires (where the length of a wire is measured in terms of the number of logic blocks it passes before being switched) and half of the programmable switches are active buffers, and half are pass transistors. In that work, however, the topology of the routing architecture prevented buffered tracks from connecting to pass-transistor tracks. This restriction prevents the creation of interconnection trees for high fanout nets that have a mixture of buffers and pass transistors. Electrical simulations suggest that connections closer to the leaves on interconnection trees are faster using pass transistors, but it is essential to buffer closer to the source. This latter effect is well known in regular ASIC routing [2].

In this work we propose a new routing architecture that allows liberal switching between buffered and pass transistor tracks. We explore various versions of the architecture to determine the density-speed trade-off. We show that one version of the new architecture results in FPGAs with 10% faster critical path delay yet uses the same area than the previous architecture that does not allow such switching. We also show that the new architecture allows a useful area-speed trade off and several versions of the new architecture result in FPGAs with 8% gain in area-delay product than the previous architecture that does not allow the switching.

1 Introduction

The routing of an FPGA consumes most of the chip area and is the dominating factor of the overall circuit delay [3]. The routing architecture of an FPGA consists of:

1. The length of each routing wire in the FPGA measured in terms of number of logic blocks that it passes before being switched.
2. The type and quantity of switches attached to each routing wire - pass transistor, multiplexor, or buffers.
3. The sizes of transistors that are used to build pass transistor and buffer switches.
4. The topology of the interconnection of the switches and routing wires in the switch blocks and connections blocks [3].
5. The routing wire width and spacing.

In this work we profile the delay properties of a previously developed routing architecture [1] in order to determine how it might be made faster and more area-efficient. The delay profile shows that some critical nets implemented in the previous architecture formed a large portion of the critical path delay because too many pass transistors were used in series. This occurred because the previous architecture had no way to easily mix buffers and pass transistors in a single source-sink connection. This drove us to design an architecture which allows buffers and pass transistors to be mixed within a single net. We will show that this new architecture delivers superior performance and density.

Several commercial architectures allow mixing of buffers and pass transistors, including the Xilinx 4000X architecture [11]. The XC4000X switch block for quad lines has one buffer and six pass transistors available, allowing the router to choose between buffering or pass-transistor connections.

This paper is organized as follows: Section 2 outlines the experimental CAD flow which is used to profile the previous architecture and produce comparisons between different routing architectures. Section 3 describes the previous routing architecture [1] and its delay profile. Section 4 describes the new architecture that allows more liberal buffer-pass transistor mixing. Section 5 presents the comparison between several versions of the new architecture and the previous architectures, and Section 6 concludes.

2 Experimental Methodology and Basic Architecture

Figure 2 illustrates the empirical methodology and CAD flow used to evaluate routing architectures. It is taken from [1][4]. Each input benchmark circuit goes through technology-independent logic optimization using SIS [6] and is technology mapped to 4-input lookup tables (LUTs) using Flowmap and Flowpack [7]. Then T-VPACK [8] is used to group 4-input LUTs and registers into “clustered” logic blocks. Following [1], we use clusters that contain four 4-LUTs each with a flip-flop, and assume a symmetric, island-style type architecture. The total number of inputs to the cluster is set at 10. VPR [4], is used to do timing-driven placement and timing-driven routing of the circuit. A brief discussion of VPR’s timing driven routing algorithms is presented in 4.1. One key output from this flow is the critical path delay of the circuit, which is determined by the timing analyzer within VPR. The routing delay is determined by calculating the Elmore delay [12] of the RC-tree network(s) of each net. Resistance arises from the wire, pass transistor resistances and tri-state buffer output resistance. Capacitance arises from the metal wires (both per unit and fringing

capacitance), and from the parasitic capacitance of the buffers and pass transistors. Under the Elmore delay model, the signal delay from the source node s_0 to destination node u is given by the following two equations:

$$d_{elmore}(s_0, u) = \sum_{e, v \in Path(s_0, u)} r_e\left(\frac{c_e}{2} + C_v\right) \quad 2\text{-}1$$

$$d_{buffer}(v, b) = d_b + r_b C_v \quad 2\text{-}2$$

c_e Capacitance of the pass transistor switch and the metal wire

r_e Resistance of the pass transistor switch e

C_v Total capacitance rooted at node v

d_b Intrinsic delay of the tri-state buffer switch b

r_b Output resistance of the tri-state buffer switch b

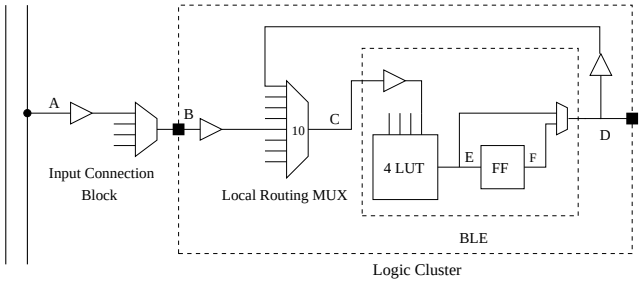

Figure 1: Speed Path of a Logic Cluster

The delay of the logic elements (LUTs, flip-flops, intra-cluster routing multiplexers) is determined by spice-level design and simulation in a 0.18um CMOS. Internal buffers and drivers are independently sized. Table 1 gives the delays of the different elements of the cluster illustrated in Figure 1 which itself is taken from [10].

The second key output from this flow is the total area required by each circuit in each architecture being evaluated. To do this, we first determine the minimum number of tracks needed to successfully route each circuit, W_{min}. Clearly this isn't possible in real FPGAs, but we believe this is meaningful as part of a logic density metric for an architecture. The router is repeatedly invoked until it finds the minimum number of tracks (W_{min}) that can route the circuit. We call this a "high stress" routing since at this track count, the circuit is barely routable. To measure the complete active area of the implementation of each benchmark circuit in each architecture, we employ the method described by Betz [1][4]. Each circuit element (e.g. LUT, multiplexor, buffer, inverter, pass transistor, configuration memory bit) is designed and properly sized at the transistor level. That is, each has been designed at spice-level and is appropriately sized to a reasonable area-delay tradeoff [4]. We measure the area of each circuit element in terms of the number of equivalent minimum-width transistors areas in the 0.18um technology. Larger transistors are counted as an appropriate number of minimum width transistors. So, once the total number of clusters is known, and the number of tracks per channel is known, the total number of minimum width transistor areas can be calculated. While this metric does not measure metal area, our communication with FPGA vendors indicates that most layouts are active-area limited [1].

It is important to note that since most designers will pick the FPGA device which has more than the minimum routing resources available, we re-route the circuit with the number of tracks per channel set to be $1.2W_{min}$. The critical path delay and the total FPGA area required are based on this so-called "low stress" routing.

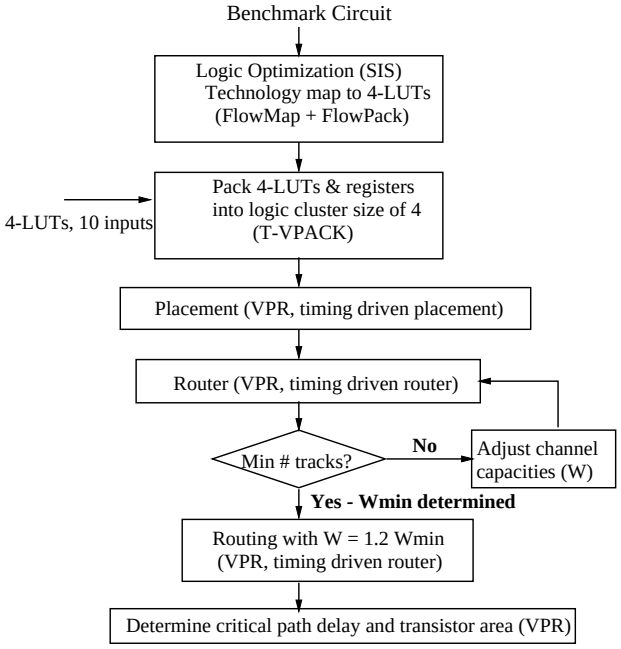

Figure 2: Architecture Evaluation Flow

3 Delay Profile of an Existing FPGA Architecture

Most of the critical path delay in FPGAs is due to routing in between logic blocks, or clusters. The first goal of this work is to identify those parts of the architecture that incur the most delay in a circuit after placement and routing. From that, we will try to improve the overall circuit speed (without sacrificing too much area) by proposing a modified architecture. We will profile the FPGA architecture proposed in [1][4], which is illustrated in Figure 3 and has the following attributes:

1. The architecture is implemented in 0.18um CMOS
2. The logic block cluster contains four 4-input lookup tables (LUTs) and flip-flops, and a total of 10 inputs and 4 outputs.
3. All routing wires span four logic blocks and have minimum-width wires.
4. Routing wire spacing is set to be double the minimum metal spacing allowed by the IC process.
5. The size of the pass transistor switch is set to be ten times the minimum-size transistor in the IC process.
6. The size of the routing buffer is set to be five times the minimum size buffer.
7. 50% of the length 4 wires are switched by pass transistors and 50% are switched by buffers.
8. The switch block employs a purely "planar" topology,

Circuit Element	Input Connection MUX (A to B)	Intra-cluster Routing MUX (B to C or D to C)	4-input LUT (C to E)	Flip-Flop Setup Time	Flip-Flop Clock to Out (E to F)
Delay (ps)	377	301	401	295	242

Table 1: Delays of Basic FPGA Circuit Elements in 0.18um CMOS

which means that once a path is connected to a pass-transistor-driven track, it can only connect to other tracks through pass transistors, and similarly for buffered tracks.

9. The flexibility of the switch block, Fs = 3 [1].
10. The flexibility of the connection block, for inputs, Fc(input) = 0.6W and Fc(output) = 0.25W, where W is the number of tracks [1].

For simplicity, only those routing switches that are connected to the left four horizontal routing tracks are shown in Figure 3. We name this routing architecture the 50-50_NO_MIX architecture, primarily to point of the percentage of pass transistors and buffers, and the fact that the two kinds of tracks cannot be inter-routed.

Figure 4 illustrates a simple routing of two nets, net A and net B in this architecture (for simplicity in the Figure we use unit-length segments rather than length 4 segments). Net A is routed by tri-state buffer switches and net B is routed by pass transistor switches.

We profile the delay of a given circuit when implemented in this architecture by measuring the portion of the total critical path delay that is attributable to the total logic block delay (delay within a cluster, including the muxing within the cluster) and the total routing delay. We further break the total routing delay into three components:

1. Source buffer delay - the delay of the buffer driving out of the logic block, and all downstream resistance and capacitance until the next buffer is encountered, either in the routing itself or at the terminating connection block. Each net has only one source buffer delay.
2. Routing buffer delay - the delay of all inside-the-routing buffers, and downstream resistance and capacitance of that buffer. Each net can have several routing buffer delays, which are summed to produce the total. If a net is routed only on pass transistor tracks then it will have zero routing buffer delay.
3. Input connection multiplexor delay - this is the delay of the multiplexors that take a net from the routing tracks into the inputs of the cluster. (A-B in Figure 1)

In Figure 4, net A has one source buffer delay, two routing buffer delays and the input connection mux delay, while net B only has source buffer delay and the input connection mux delay since routing buffer switches are not used in the routing.

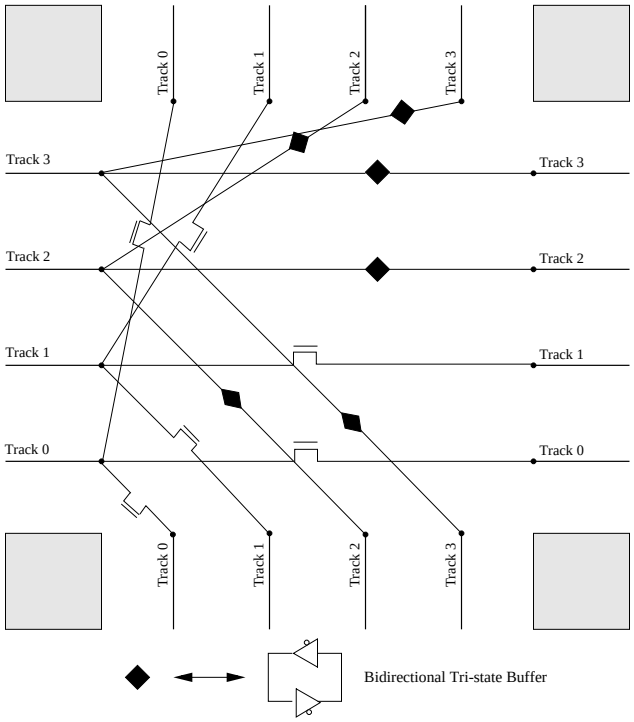

Figure 3: 50-50_NO_MIX Switch Block

3.1 Profile of 50-50_NO_MIX Routing Architecture

The delay profile for the 20 largest MCNC circuits [9] implemented in the 50-50_NO_MIX routing architecture was calculated as described above. Table 2 gives the profile; the first column and the second column give the circuit name and its size in terms of the number of 4-input BLEs respectively. The third column gives the total critical path delay while the fourth column gives the percentage of the total delay due to the logic cluster. The fifth column gives the percentage of delay due to the extra-cluster routing delay, and the sixth column gives the percentage of total delay due to the source buffer alone. The seventh column gives the percentage delay due to the routing buffers. The eighth column gives the percentage delay due to the input routing multiplexer. The second last row gives the geometric average of each column and the last row gives the arithmetic average.

Notice that the “Routing Buffer Delay” column is 0% in four cases. This occurs because the entire critical path is routed on

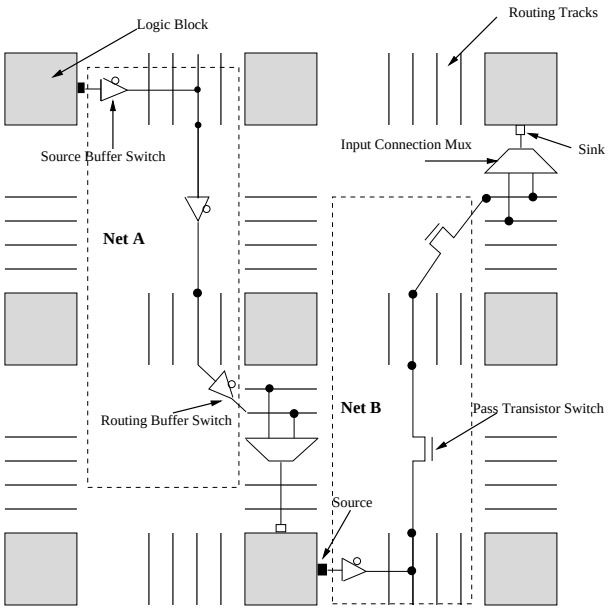

Figure 4: Inter-cluster Routing of Net A and Net B

tracks that use only pass transistors, and therefore exhibit no routing buffer delay.

The key item to observe from the profile is that the source buffer delay accounts for more than 50% of the routing delay on average. When this delay is large, most of this delay comes from the source buffer driving pass-transistor only routing trees, because in the 50-50_NO_MIX architecture, once the source buffer starts driving a pass transistor track, it isn't possible to switch to drive a buffered track, or vice-versa. This can and does result in a large pass-transistor only RC network, which exhibits quadratic delay in the distance traversed. The reader may question why this happens with a timing-driven router that should prevent the use of slow resources for critical nets. The reason critical nets are assigned to slower resources is that even more critical nets *are* given the fast resources, and they are all used up.

One solution is to provide more buffered tracks that the 50% number in the 50-50_NO_MIX architecture. Appendix A shows results of different percentage of mix between buffers and pass transistors for non-mix architectures. There are two reasons this solution is not good: 1. Buffers are far more expensive in silicon area than pass transistors, and so this would cost a great deal. 2. Pass transistors are *faster* for shorter connections; removing them means that critical nets that travel a short distance will be less likely to achieve good speed.

An alternative is to architect a routing fabric that allows liberal switching between tracks that are switched by pass transistors and tracks that are switched by buffers. This would permit the use of buffered connections near the source of a large fanout tree, and pass transistors near the destination, which is the best of both worlds. We propose such an architecture in the next section.

4 A New Architecture That Allows Pass Transistor and Buffer Mixing

We present a new routing architecture, called the Mixed Buffer-Pass routing architecture, which allows routes to switch between buffers and pass transistors within a single source-sink connection. Figure 5 illustrates the Mixed Buffer-Pass routing architecture. Note that, for simplicity, Figure 5 shows only the programmable connectivity of the wires entering the switch block from the left hand side. This architecture divides the routing tracks into the following three classes of track:

1. *Straight-Planar:* These tracks are switched using pass transistors, in the planar switch topology and have *no* programmable connections to the other two classes described below.
2. *Mixed-Buffer:* These tracks programmably connect to each other using tri-state buffer switches in a planar (Fs=3) topology, and connect to the Mixed Pass tracks (described below) using pass transistors.
3. *Mixed-Pass:* These tracks programmably connect to each other using pass transistor switches in a planar (Fs=3) topology, and also connect to the Mixed Buffer tracks (described above) using pass transistors.

The Mixed-Buffer and Mixed-Pass transistors tracks are present in equal numbers in each channel so as to allow the creation of a simple pattern of interconnection between them. The connections between the Mixed-Buffer and Mixed-Pass transistor tracks are two additional pass transistors per track that occur on every track at the point at which it switches. These are illustrated by the **bold** transistors in Figure 5. While regular tracks typically switch to three directions upon termination at a switch block, these two additional switches allow each track to turn in the left and right directions - there is no additional "straight" connection using a pass transistor in order to reduce area cost and capacitive loading of the mixed buffer-pass tracks.

These Mixed-pass transistor tracks are more expensive, and are slightly more loaded (by the extra switches) than the straight-planar tracks. This is the reason that we have included the straight-planar tracks - they are cheaper and somewhat faster than the Mixed-Pass tracks. Experiments presented below will explore the appropriate portion of each of the three classes of tracks.

4.1 Timing-Driven Routing Algorithm

In this section we discuss a relevant feature of the timing-driven routing algorithm that is used to exploit the mixed buffer and pass-transistor tracks. We use the timing-driven router in VPR [4] which is based on the Pathfinder negotiated congestion router [13]. We will not describe the VPR router in detail but instead refer the reader to [4]. Briefly, it employs a directed maze-type expansion which uses a node costing function that accounts for congestion and delay. There are two portions of the delay calculation: 1. The determination of the delay from the source to the current wavefront expansion point. This delay can be calculated exactly because the resistance and capacitance to this point is exactly known. 2. The estimation of the delay from the current wavefront expansion point to the target sink. Since the routing isn't complete, this RC network is unknown. The VPR router assumes that the subsequent route will use routing resources identical to the type employed at the wavefront point. For example, if the current wavefront point is a buffered segment of length 4 then the router assumes, for purposes of calculating forward-looking delay, that the entire remainder of the route will consist of buffered segments

Circuit Name	# of 4-Input BLEs	Total Critical Path Delay (ns)	Breakdown of Total Delay		Breakdown of Routing Delay		
			Logic Block Delay (%)	Routing Delay (%)	Source Buffer Delay (% of Routing Delay)	Routing Buffer Delay (% of Routing Delay)	Input Connection MUX Delay (% of Routing Delay
alu4	1522	13.4	38.8	61.2	67.8	0	32.2
apex2	1878	15.1	34.4	65.6	73.8	3.4	22.9
apex4	1262	18.4	24.4	75.6	75.4	11.0	13.5
bigkey	1707	7.8	28.6	71.4	86.4	0	13.5
clma	8383	29.9	26.3	73.7	64.7	24.8	10.3
des	1591	11.6	32.6	67.4	76.0	0	24.0
diffeq	1497	16.3	61.1	38.9	34.7	0	65.3
dsip	1370	5.9	38.1	61.9	59.6	9.4	31.0
elliptic	3604	19.7	54.3	45.7	33.0	24.9	42.0
ex1010	4598	30.7	16.9	83.1	80.3	10.7	8.9
ex5p	1064	12.6	35.6	64.4	53.4	18.6	27.8
frisc	3556	24.4	63.8	36.2	40.6	3.9	55.5
misex3	1397	14.2	31.7	68.3	65.7	11.0	23.2
pdc	4575	36.1	14.4	85.6	83.6	7.8	8.5
s298	1931	30.5	32.6	67.4	75.5	2.5	22.0
s38417	6406	15.9	49.3	50.7	37.9	10.7	51.3
s38584	6447	12.6	52.5	47.5	35.8	20.0	44.2
seq	1750	12.2	37.0	63.0	55.2	15.4	29.5
spla	3690	22.8	22.8	77.2	82.5	4.5	12.8
tseng	1047	14.9	62.1	37.9	33.2	0	66.8
Geomet-ric Average	2390	16.54	35.0	62.0	57.6	--	25.1
Arith-metic Average	2964	18.25	37.9	62.1	60.7	8.9	30.3

Table 2: Critical Path Delay Distribution of the 50-50_NO_MIX Architecture

of length 4. Similarly, if the segment was unbuffered, the forward-looking delay estimator would assume all subsequent segments to the sink would be connected with pass transistors. Clearly this approach is more accurate for non-mixed architectures. However, since in mixed architectures the forward-looking route is truly unknown, there is no better guess to make. Also, once any future point is reached, the exact calculation (described in 1 above) is correct. Depending on how directed the router is the search will be either more breadth-first or depth-first. The breadth-first expansion will be more accurate, as it always has the most correct delay calculation. Our empirical experience has shown that the VPR router does sufficient breadth-first searching to achieve a good quality answer. It typically uses buffers near the source of high-fanout nets and pass transistors close to the sink in the mixed architectures.

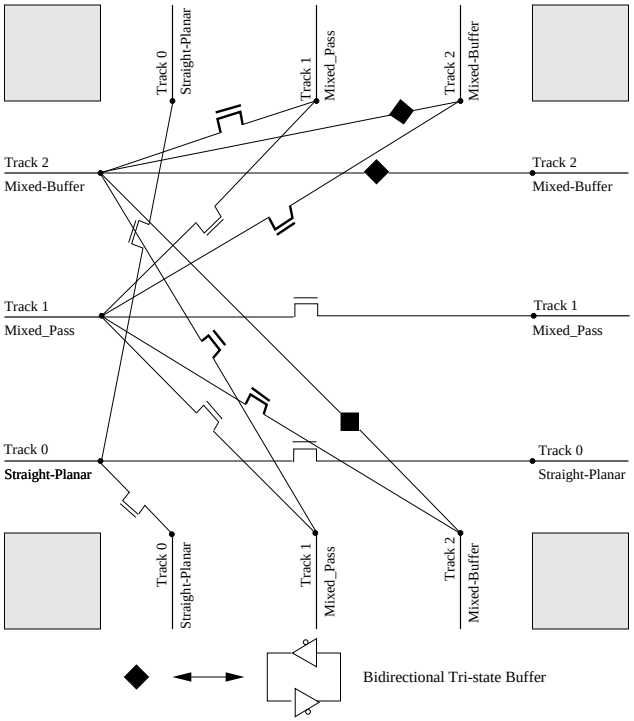

Figure 5: Mixed Buffer-Pass Switch Block

5 Experimental Results

In this section we explore which proportion of each type of track described in Section 4 provides the best area, delay and area-delay product for the new architecture. We use the same set of benchmark circuits, the 20 largest MCNC circuits, as the circuits we use to profile the 50-50_NO_MIX routing architecture in Section 3. We first present four figures of merit (track count, the critical path delay, total area and area-delay) of the new Mixed Buffer-Pass architecture as a function of the percentage of straight-planar tracks. Then we compare the figures of merit of several versions of the new architecture to several versions of the non-mixed architecture.

5.1 Properties of the Mixed Buffer-Pass Routing Architecture

Figure 6 plots the geometric average, over 20 circuits, of the minimum number of tracks per channel required to successfully route each circuit as a function of the percentage of straight-planar tracks. For routing architectures with low percentage of straight-planar tracks, the total track count is lower because of the increased flexibility between mixed-buffer and mixed-pass tracks provided by the two additional switches. When there is a high percentage of straight-planar tracks, there is a slightly increase in track count. This is likely because the timing-driven router [4] is forced to route nets in a more star-like pattern in order to achieve reasonable timing, which uses up more tracks that, for example, a steiner tree.

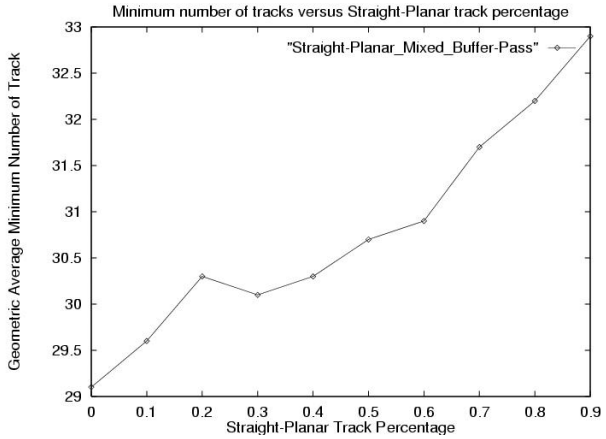

Figure 6: Geometric Average Track Counts for Mixed Buffer-Pass Transistor Architecture

Figure 7 is a plot of the geometric average of the total area (as described in Section 2) which includes the logic area and routing area (for all 20 circuits) versus the percentage of straight-planar tracks. The total area decreases as the percentage of the mixed-buffer tracks decreases because tri-state buffer switches consume twice as much area as pass transistor switches [1]. In addition mixed-pass tracks are more expensive than straight-planar tracks in terms of area because of the extra switches attached. Note that the track count increase observed in Figure 6, is not sufficient to offset the significantly higher area of mixed-buffer and mixed-pass tracks. Observe also that the use of mixed-buffer and pass tracks causes a significant increase in total area, over 40% more area compared to 100% pure planar tracks.

Figure 8 is a plot the geometric average of the critical path delay as a function of the percentage of straight-planar tracks. As the percentage of the mixed buffer-pass tracks decreases, the critical path delay increases. This is expected when there is not enough buffered routing resources to route high fanout nets.

Figure 9 is a plot of the geometric average of the total area delay product as a function of the percentage of straight-planar tracks. Notice that the total area delay product reaches its minimum when the percentage of the straight planar tracks is approximately 70%.

Comparison and Architecture		Area (x10^6 min width transistor)	Critical Path Delay (ns)	Area-Delay Product
Comparison 1 (Best Delay)	Mixed Buffer-Pass Routing Architecture (20% Straight Planar tracks, 80% Mixed Buffer-Pass tracks)	5.38	14.94	6.51
	NO_MIX Routing Architecture (20% pass transistor tracks, 80% buffered tracks)	6.20	14.31	7.33
Comparison 2 (Best Area-Delay Product)	Mixed Buffer-Pass Routing Architecture (70% Straight Plane tracks, 30% Mixed Buffer-Pass tracks)	4.35	17.48	5.72
	NO_MIX Routing Architecture (80% pass transistor tracks, 20% buffered tracks)	4.38	18.89	6.25
Comparison 3 (Best new vs. 50_50 NO_MIX)	Mixed Buffer-Pass Routing Architecture (20% Straight Plane tracks, 80% Mixed Buffer-Pass tracks)	5.38	14.94	6.51
	NO_MIX Routing Architecture (50% pass transistor tracks, 50% buffered tracks)	5.25	16.54	6.90

Table 3: Comparisons Between Mixed and Non-mixed Architectures

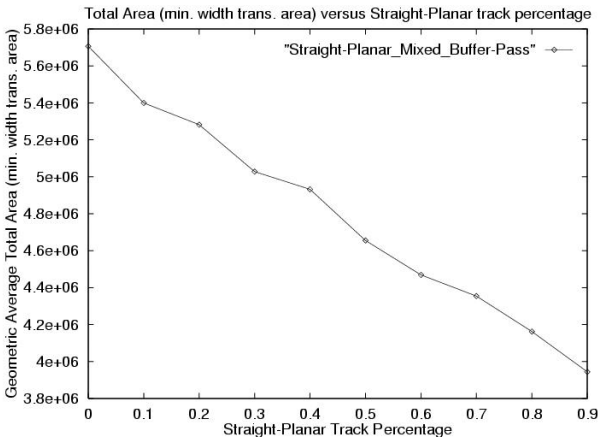

Figure 7: Total FPGA (Routing + Logic Block) Area (low stress routing) for Mixed Buffer-Pass Transistor Architecture

5.2 Comparison of Mixing and Non-Mixing Architectures

In this Section we compare several versions of the Mixed Buffer-Pass routing architecture (with differing amounts of straight-planar tracks) to several versions of a non-mixing routing architectures (which have different amounts of pass transistor tracks). Table 4 summarizes the comparison. Appendix A provides the plots of track count, area, critical path delay and area-delay product of the non-mixed architecture as a function of the percentage of pass-transistor tracks in the same 0.18um CMOS process. (Note that [1] and [4] work in 0.35um).

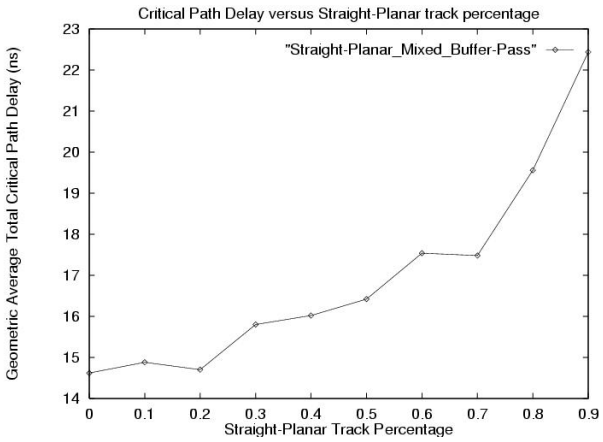

Figure 8: Critical Path Delay for Mixed Buffer-Pass Transistor Architecture

The first comparison ("Comparison 1" in Table 4) is between the fastest mixed architecture and the fastest non-mixed architecture. The Mixed Buffer-Pass architecture with 20% Straight-Planar tracks achieves almost the same critical path delay (just 4% higher) achieved by a non-mixed architecture with 80% buffered tracks, yet uses 13% less area. The new architecture results in 11% gain in area-delay product.

The second comparison ("Comparison 2" in Table 4) is between the architectures that achieve the best area-delay product for the mixed and non-mixed architectures. The Mixed Buffer-Pass architecture with 70% Straight-Plane track percentage consumes the same area consumed by the non-mixed architecture with 20% buffered tracks, yet results in 8% faster in critical path delay. The new architecture results in 8% gain in area-delay product.

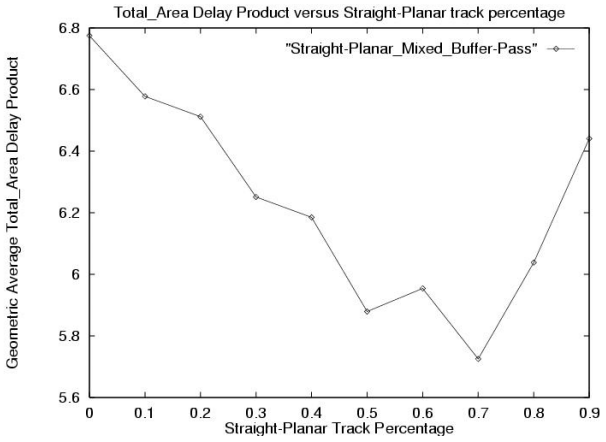

Figure 9: Area-Delay Product for Mixed Buffer-Pass Transistor Architecture

The third comparison is between a mixed architecture with 20% Straight-Planar tracks and the 50-50_NO_MIX architecture selected in [1]. The new architecture consumes almost the same area (just 2.4% more) consumed by the 50-50_NO_MIX architecture, yet results in 10% faster in critical path delay on average. The new architecture results in 6% gain in area-delay product.

5.3 Delay Profile of the Mixed Buffer-Pass Architecture

Table 4 gives the delay profile of a version of the Mixed Buffer-Pass architecture with 0% straight-planar tracks. Compared to the 50-50_NO_MIX delay profile presented in Table 2 this architecture is, on average 11.6% faster. Observe also that the percentage of delay attributed to the source buffer is significantly reduced. For each circuit, the speed gain (between this mixed architecture and the 50-50_NO_MIX architecture) ranges from -6.1% for the circuit *elliptic* to +47.1% for the circuit *pdc*. Notice that if circuit does not have many high fan-out nets, the benefits of the new architecture diminish. This is because that the new architectures pays the price of adding two pass transistor switches per mixed-buffer track and therefore increase the capacitive loading for each mixed-buffer track. If the number of high fan-out nets is relatively small, straight-planar tracks are very effective to route low fan-out nets.

Circuit Name	# of 4-Input BLEs	Total Critical Path Delay (ns)	Breakdown of Total Delay		Breakdown of Routing Delay		
			Logic Block Delay (%)	Routing Delay (%)	Source Buffer Delay (% of Routing Delay)	Routing Buffer Delay (% of Routing Delay)	Input Connection MUX Delay (% of Routing Delay
alu4	1522	11.8	44.2	55.8	19.5	40.3	40.2
apex2	1878	14.2	41.5	58.5	29.2	43.4	27.2
apex4	1262	11.9	32.0	68.0	40.4	31.5	28.1
bigkey	1707	6.8	32.8	67.2	21.4	54.0	24.6
clma	8383	25.9	41.2	58.8	19.9	50.3	29.8
des	1591	11.6	38.9	61.1	34.2	33.7	32.1
diffeq	1497	16.4	60.7	39.3	35.9	0	64.1
dsip	1370	6.0	37.2	62.8	21.5	48.6	29.9
elliptic	3604	20.9	24.1	75.9	17.4	65.9	16.6
ex1010	4598	17.4	29.9	70.1	13.7	61.5	24.8
ex5p	1064	12.9	40.2	59.8	23.9	42.0	34.1
frisc	3556	25.1	62.1	37.9	42.5	5.8	51.5
misex3	1397	13.6	33.1	66.9	11.4	67.9	20.3
pdc	4575	19.1	30.9	69.1	18.7	61.4	20.0

Table 4: Critical Path Delay Distribution of Mixed Buffer-Pass Architecture

Circuit Name	# of 4-Input BLEs	Total Critical Path Delay (ns)	Breakdown of Total Delay		Breakdown of Routing Delay		
			Logic Block Delay (%)	Routing Delay (%)	Source Buffer Delay (% of Routing Delay)	Routing Buffer Delay (% of Routing Delay)	Input Connection MUX Delay (% of Routing Delay
s298	1931	23.7	45.0	55.0	53.3	9.1	37.6
s38417	6406	16.1	48.9	51.1	33.9	24.8	41.3
s38584	6447	12.3	53.7	46.3	34.8	19.0	46.3
seq	1750	12.1	37.1	62.9	30.8	44.4	24.6
spla	3690	18.6	31.8	68.2	33.0	46.1	20.8
tseng	1047	15.4	60.3	39.7	35.1	3.1	61.7
Geometric Average	2390	14.62	39.9	57.6	26.5	--	31.5
Arith-metic Average	2964	15.59	41.3	58.7	28.5	37.6	33.8

Table 4: Critical Path Delay Distribution of Mixed Buffer-Pass Architecture

6 Conclusions

We have shown the importance of mixing tri-state buffer switches and pass transistor switches which make up the inter-cluster routing connections. The routing architectures which allow router to choose switch type during the routing phase are faster or more area-efficient. A version of the new architecture with 20% straight-planar tracks and 80% mixed buffer-pass tracks results in 10% gain in speed without area penalty or 13% gain in area without critical path delay penalty compared to a non-mixed routing architecture.

7 Acknowledgments

We would like to thank Vaughn Betz and Alexander Marquardt for providing us the CAD framework, VPR, upon which our work is built. We would also like to express special thanks to Elias Ahmed for providing us the timing information of the logic blocks for the 0.18 um CMOS technology.

8 References

[1] V. Betz, J. Rose, “FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density”, FPGA’99, pp. 59-68.

[2] J. Cong, K.S. Leung, “Optimal Wiresizing Under the Distributed Elmore Delay Model,” Proc. Intl’l Conf. on Computer Aided Design, 1993, pp. 634-639.

[3] S. Brown, R. Francis, J. Rose, Z. Vranesic, “Field-Programmable Gate Arrays”, Kluwer Academic Publishers, 1992.

[4] V. Betz, J. Rose, A. Marquardt, “Architecture and CAD for Deep-Submicron FPGAs”, Kluwer Academic Publishers, 1999.

[5] E. Ahmed, J. Rose, “The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density”, FPGA’00, Monterey, CA, 2000.

[6] E.M. Sentovich et al, “SIS: A System for Sequential Circuit Analysis”, Tech. Report No. UCB/ERL M92/41, University of California, Berkeley, 1990.

[7] J. Cong, Y. Ding, “FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs”, IEEE Trans. on CAD, Jan. 1994, pp.1-12.

[8] A. Marquardt, V. Betz, J. Rose, “Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density”, ACM/SIGDA FPGA 99, 1999.

[9] S. Yang, “Logic Synthesis and Optimization Benchmarks, Version 3.0”, Tech. Report, Microelectronics Centre of North Carolina, 1991.

[10] E. Ahmed, “M.A.Sc Thesis”, University of Toronto, 2000.

[11] Xilinx XC4000X Data Book, May 14, 1999.

[12] W. Elmore, “The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers,” Journal of Applied Physics, January 1948, pp. 55-63.

[13] C. Ebeling, L. McMurchie, S. A. Hauck and S. Burns, “Placement and Routing Tools for the Triptych FPGA,” IEEE Trans. on VLSI, Dec. 1995, pp. 473-482.

Appendix A: Experimental Results for 0.18um Non-Mixed Architectures