FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density

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

Abstract

In this work we investigate the routing architecture of FPGAs, focusing primarily on determining the best distribution of routing segment lengths and the best mix of pass transistor and tri-state buffer routing switches. While most commercial FPGAs contain many length 1 wires (wires that span only one logic block) we find that wires this short lead to FPGAs that are inferior in terms of both delay and routing area. Our results show instead that it is best for FPGA routing segments to have lengths of 4 to 8 logic blocks. We also show that 50% to 80% of the routing switches in an FPGA should be pass transistors, with the remainder being tri-state buffers. Architectures that employ the best segmentation distributions and the best mixes of pass transistor and tri-state buffer switches found in this paper are not only 11% to 18% faster than a routing architecture very similar to that of the Xilinx XC4000X but also considerably simpler. These results are obtained using an architecture investigation infrastructure that contains a fully timing-driven router and detailed area and delay models.

1 Introduction

FPGAs consist of a large number of programmable logic blocks, which can each implement a small amount of digital logic, and programmable routing which allows the logic block inputs and outputs to be connected to form larger circuits. The delay of a circuit implemented in an FPGA is mostly due to routing delays, rather than logic block delays, and most of an FPGA's area is devoted to programmable routing [1]. Furthermore, as FPGAs move into increasingly deep submicron IC processes, the fraction of total delay due to routing is increasing with each process generation [2]. Consequently, one must devise routing architectures which are both fast and area-efficient to create an FPGA that fully exploits the performance and density potential of deep-submicron technologies.

In this paper we investigate island-style FPGA routing architectures; the FPGAs of Xilinx [3], Lucent Technologies [4], and Vantis [5] employ this style of routing architecture. A simplified view of an island-style FPGA is shown in Figure 1. The routing architecture of an FPGA defines such features as:

1. The length of each routing wire segment (how many logic blocks a routing wire spans before terminating),
2. Whether each routing switch is a pass transistor or a tri-state buffer,
3. Where routing switches are located and which routing wires they can connect together,
4. Which routing wires in the channel adjacent to a logic block input or output can connect to that logic block pin,
5. The sizes of the transistors used to build the various programmable switches, and
6. The metal width and spacing of the routing wires.

In Figure 1, for example, half the routing tracks consist of length 1 wire segments, while the other half consist of length 2 wire segments. Some of the programmable routing switches are pass transistors, while others are tri-state buffers.

In this paper we will focus primarily on determining the best values for parameters 1 and 2 above: the best *segmentation distribution* (lengths of routing wire segments) and the best mix of pass transistor and tri-state buffer switches. We have investigated appropriate values for the other four parameters [6, 7], but space limitations preclude more than a brief discussion of these other four issues in this paper. Note however, that we set the other 4 parameters to reasonable values throughout the experiments of this paper, as this is essential for meaningful architectural comparisons.

Routing architecture design is very challenging because the best value for each of the parameters above depends on complex trade-offs. For example, in an FPGA with too many short wires, some long connections will be constructed using several short wire segments connected in series, resulting in poor speed. If an FPGA includes too many long wires, however, some short connections will be forced to use long wire segments, degrading speed and wasting area. Similarly, an architecture with too many or too few tri-state buffer routing switches will likely be suboptimal. Pass transistor switches require less area, and they are faster than buffers for short connections, but connections that pass through many

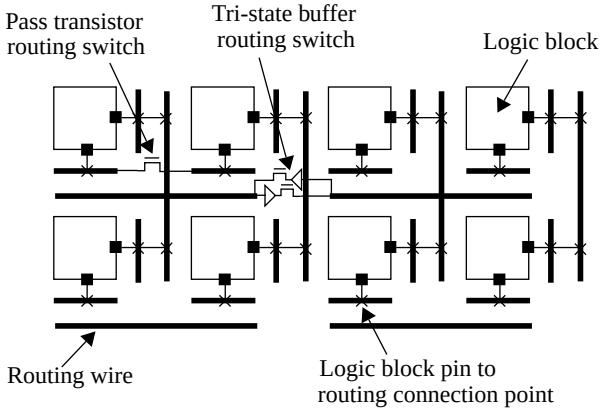


Figure 1: Example island-style FPGA routing architecture.

series switches are better served by tri-state buffers. As well, the best mix of routing switches is dependent on the length of the FPGA's routing wire segments. In an FPGA with many long wires, it will rarely be necessary to connect many switches in series to make a connection. Consequently, such an FPGA can likely use a higher fraction of pass transistor-based switches in its routing than an FPGA that contains few long wires.

Considering the importance of routing architecture to both FPGA area and speed, relatively little research has been conducted in this area. Some prior work [8, 9, 10, 11] has investigated different switch topologies for use in FPGAs where all wire segments are of length 1 (i.e. span only a single logic block) and has compared architectures only in terms of area-efficiency. Some studies have investigated the best distribution of routing wire segment lengths for use in row-based FPGAs [12, 13, 14, 15], but the methodology used in these studies is not applicable to island-style FPGAs. The most directly comparable work to this paper is that of Brown et al [16, 17] and Chow et al [18]. These studies investigated island-style FPGAs which contained some longer wire segments. We extend this prior research in several important ways, however. First, we consider the possibility of some routing switches being tri-state buffers, whereas all prior research has investigated FPGAs containing only pass transistor switches. Second, we compare FPGAs on the basis of the "true" delay metric — critical path delay of benchmark circuits — and a detailed, transistor-based area model; prior research has used simpler, and less accurate, delay and area metrics. Third, the CAD flow we use to evaluate architectures employs a combined global and detailed (one-step) router, while Brown et al and Chow et al performed global and detailed routing in two steps. Since the global router in a two-step routing is unaware of the distribution of wire segment lengths, and hence not attempting to optimize for it, long wires may not be used as effectively as possible. Finally, the router we use in this study is fully timing-driven (uses timing analysis to determine which connections need high speed routing) allowing it to more fully exploit the intrinsic speed of different routing architectures, and hence improving the accuracy of our architecture comparisons.

The organization of this paper is as follows. The next section describes the portions of the FPGA architecture which are held constant throughout the experiments of this paper. Section 3 then outlines the experimental framework we use to compare different FPGA routing architectures. In Section 4, we perform experiments to determine which length of wire segment results in the best speed and area-efficiency when all the routing wires in an FPGA have the same length, and all routing switches are tri-state buffers. In Section 5 we investigate more complex routing architectures that contain wire segments of two different lengths, and a mix of pass transistor and tri-state buffer routing switches. Section 6 compares some of the best architectures we have found to a routing architecture similar to that of the Xilinx 4000X series FPGAs. Finally, we summarize our results and conclusions.

2 FPGA Architecture and Circuit Design Parameters Held Constant

In this paper we are investigating different routing architectures, so we hold the other architectural parameters, such as the logic block used, constant throughout the experiments. In all our experiments, each channel in an FPGA contains the same number of tracks and has the same segmentation distribution.

The logic block of all the FPGAs studied in this work is a logic cluster [19] of four 4-input look-up tables (4-LUTs) and registers, with ten inputs, four outputs, and one clock. This logic block includes local routing that allows each of the LUT inputs to be connected to any of the 10 logic block inputs or any of the four outputs generated within the logic block.[1] This logic block is more typical of the size of current commercial FPGA logic blocks than the single 4-LUT logic block assumed by most prior routing architecture research, and prior research has shown that this logic block leads to an area-efficient FPGA [19]. The input and output pins are evenly distributed around the perimeter of the logic block, since [20] showed that this pin positioning is best.

The number of I/O pads that fit into the height or width of a logic block is set to four, in line with the relative sizes of pads and 4-LUTs of current FPGAs [3, 4, 5, 21]. With this assumption three of the twenty benchmark circuits we use (bigkey, des and dsip) are pad-limited. We always map each circuit to the smallest square logic block array that has enough logic blocks and pads to accommodate it. Since commercial FPGAs normally distribute the circuit clock through a special, dedicated routing resource, we do not route the clock net in sequential circuits.

The switch block topology [8] (which defines which routing wire segments can connect via routing switches at the intersection of a horizontal and vertical channel) used throughout this paper is the *disjoint* switch block topology used in the original Xilinx 4000 series FPGAs [22]. In this switch block, a wire in track number *i* can connect only to other wires in track *i*. Note that with this switch block wires of a given length can only connect to other wires of the same length. Interestingly, while the disjoint switch block topology is not as routable as the Wilton switch block topology in FPGAs where all routing wire segments have length 1 [11], we have found that it results in better speed and area-efficiency than a straightforward generalization of the Wilton switch block in FPGAs that contain longer wires [6, 7]. Previous switch block research has focused exclusively on FPGAs containing only length 1 wire segments; clearly future research should investigate the interplay between segmentation distribution and switch block topology.

We set the number of routing tracks to which each logic block pin can connect, F_c [8], to 0.5W, where W is the number of routing tracks in a channel. Our experiments have shown that this is a good value for a wide range of routing architectures.

The size of the transistors used in the routing switches is a key architectural issue. The metal capacitance of a routing wire segment is quite large, so significant speed improvements can be achieved by increasing the size of the transistors forming pass transistor or tri-state buffer routing switches. At some point, however, one achieves diminishing speed returns as the size of the switch transistors is increased, since the parasitic capacitance added by these switches becomes comparable to the metal capacitance. As well, increasing the size of the routing switch transistors requires more layout area. Essentially we want to choose the transistor size that achieves the best trade-off between the speed of the routing and the area required. We believe that the best trade-off occurs when the transistor sizes are chosen to minimize the area-delay product of the resulting routing resources. We evaluated the speed of FPGA routing structures using different transistor sizes in TSMC's 0.35 µm, three-level metal CMOS process [23], and we used the area model described in Section 3.3 to evaluate the area required. We found that for a very wide range of wire segment

1. Since this local routing is part of the logic block, and not part of the "general" FPGA routing, we count its area as part of the logic block area, rather than as part of the FPGA routing area, in the later sections.

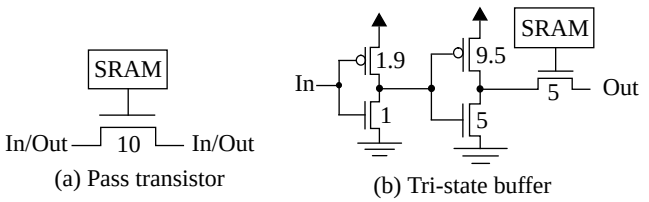


Figure 2: Transistor widths used to build routing switches.

lengths, pass transistor routing switches achieve the best area-delay product when they are ten times as wide as the minimum contactable transistor width. Similarly, we found that for a wide range of wire segment lengths the tri-state routing buffer with the best area-delay product is the two-stage buffer shown in Figure 2; its output stage has five times the minimum drive strength. Note that the transistor widths in Figure 2 are all in "times minimum contactable width" rather than in microns. For details on the experiments we performed to determine these best routing transistor sizes, see [6, 7].

Finally, we have to choose the metal width, spacing and layer in which routing wires are laid out. Throughout this paper we assume routing wires are laid out in metal 3; we have found that our results do not change significantly if routing wires are laid out in metal 1 or 2, however. We also use minimum-width metal for all routing wires. While increasing the metal width reduces the metal resistance, it also increases the metal capacitance. Since the resistance of a routing switch is considerably larger than the metal resistance of even fairly long routing wire segments, we have found that the net effect of wider metal wires is to increase the routing delay. Throughout this paper we also assume routing wires use the minimum metal spacing; this results in the highest wiring density during layout, at the cost of higher metal capacitance than wider metal spacings would yield. We have also run many of the experiments in this paper with wider metal spacings, however. While this increases the speed of *every* FPGA architecture (by about 15%), it does not change any of our architectural conclusions.

3 Experimental Methodology

We explore different architectures by implementing the twenty largest MCNC benchmark circuits [24] into each FPGA architecture of interest. These circuits range in size from 1064 to 8381 4-LUTs. We implement each circuit with an automatic CAD flow similar to that used by FPGA users: technology-independent logic optimization, technology-mapping, placement and routing. We then compare the circuit delay achieved and the area required by each architecture.

3.1 CAD Flow

Figure 3 illustrates the CAD flow we use to evaluate routing architectures. First, SIS [25] is used to optimize the logic of each circuit. Next, we use Flowmap [26] to technology map each circuit into 4-LUTs and registers, and Flowpack [26] to optimize the mapping and reduce the number of LUTs required. VPack [27] then groups these 4-LUTs and registers into logic blocks of the desired size (clusters of at most four LUTs, using no more than 10 distinct inputs). Versatile Place and Route (VPR) then places the circuit [28], and the VPR timing-driven router [6, 7] is repeatedly invoked with different channel capacities to determine the minimum number of tracks per channel, W_{min}, required to route the circuit. FPGA manufacturers normally build enough routing into their FPGAs that "average" circuits have some spare routing available. We model this by performing a final "low-stress" routing of each circuit with the number of tracks per channel set to $1.2 \cdot W_{min}$. Our delay model then estimates the circuit critical path, and our area model estimates the total transistor area needed to lay out all the routing in this FPGA. At the end of this CAD flow, then, we have enough information to compare both the speed and the area-efficiency of one architecture to another.

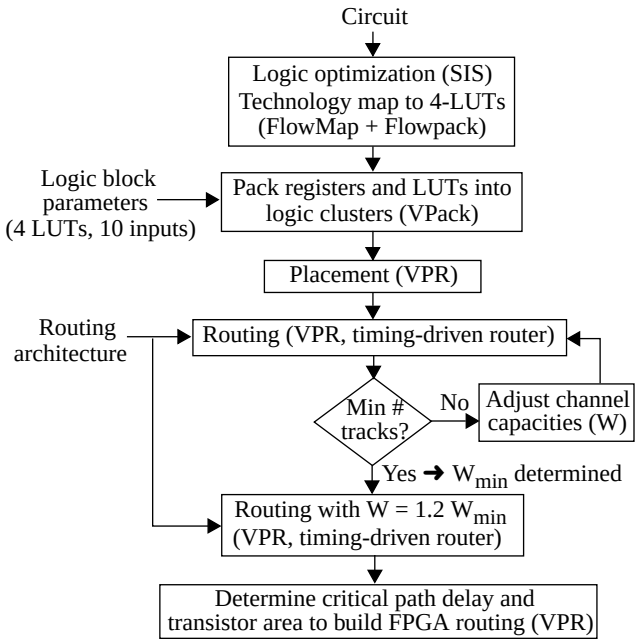


Figure 3: Architecture evaluation flow.

3.1.1 Overview of Timing-Driven Routing Algorithm

Since we are comparing the speed and area of different FPGA routing architectures, the most important tool in the CAD flow above is the router. To allow fair comparisons of different routing architectures, we created a new router that optimizes well for both speed and routability, and can fully exploit the features of each FPGA architecture we study.

Like its purely routability-driven predecessor [28], the VPR timing-driven router [6, 7] uses many ideas from the Pathfinder routing algorithm [29]. It repeatedly rips-up and re-routes every net in the circuit, and gradually resolves routing congestion by gradually increasing the cost of overused routing resources. Like Pathfinder, it also performs timing analysis repeatedly during routing, and uses the slack of each connection to determine the congestion avoidance / delay minimization trade-off to use for that connection. The VPR timing-driven router contains some significant new features, however. The most important enhancement is that this router directly optimizes the Elmore delay [30] as it routes each connection. The router uses the Elmore delay to guide it as it selects the net topology, wire segment lengths, and the type of switch (pass transistor or tri-state buffer) used to connect two wire segments. Note also that it makes all these decisions in one unified step, since the limited flexibility of FPGA routing means that topology, wire segment length and switch type decisions are all coupled. Previous academic FPGA routers have optimized either only wirelength or the linear delay model, in which each routing

switch has a fixed delay.[1] The delay of a pass transistor is highly dependent on the topology of the net containing it, however, so by optimizing the Elmore delay our router is able to make better net topology choices and determine when using a buffered routing switch is preferable to using a pass transistor.

This new router requires only slightly (6% on average) more tracks than the VPR routability-driven router to successfully route circuits. Since the VPR routability-driven router requires fewer tracks to successfully route a set of standard benchmarks than any other router for which results are available [28], this implies the new timing-driven router optimizes for routability very well. In addition, the VPR timing-driven router produces circuits which are *2.6 times* faster, on average, than those produced by the VPR routability-driven router. Clearly timing-driven routing is essential to obtain good circuit speed.

3.2 Delay Model

Our delay values are all based on the delays in TSMC's 0.35 µm, 3.3 V CMOS process. To determine the critical path of a circuit, we must:

1. Determine the delay of every connection internal to a logic block,
2. Determine the delay of every connection between logic blocks, and
3. Perform a path-based timing analysis of the circuit using these delay values.

We found the delay of the connections within logic blocks by performing SPICE simulations of every structure in a logic block. See [6, 7] for transistor-level schematics of the logic block we use, and a listing of various important delays.

After a routing is complete, we can determine the delay of every routed connection. We model pass transistors and buffers by equivalent circuits composed of resistors, capacitors, and idealized, constant delay elements. The values of the various equivalent resistances, capacitances and buffer intrinsic delays were determined from SPICE simulations with the TSMC 0.35 µm process. After a routing is complete, VPR uses these simplified models of pass transistors and buffers, as well as metal capacitance and resistance data, to build an equivalent RC-tree for each net. It then computes the Elmore delay from the source to each of the sinks. We have found the accuracy of this delay model to be quite good — the connection delays it predicts are almost always within 9% of SPICE [6, 7].

Finally, VPR performs a path-based timing-analysis [32] using these connection delay values to determine the circuit critical path.

3.3 Area Model

Since the area of typical commercial FPGAs is dominated by transistor area,[2] the most accurate way to assess the area of an FPGA architecture, short of actually laying out each FPGA architecture studied, is to estimate the total transistor area required by its layout. Our area model is based on counting the number of *minimum-width transistor areas* required to implement each FPGA architecture. A minimum-width transistor area is simply the layout area occupied by the smallest transistor that can be contacted in a process, plus the minimum spacing to another transistor above it and to its right. By counting the number of minimum-width transistor areas required to implement an FPGA, rather than the number of square microns which these transistors would occupy, we obtain a process-independent estimate of the FPGA area. Transistors larger than minimum-width are counted as several minimum-width transistor areas. VPR computes the routing area of an FPGA by "building" every structure required by the FPGA's routing, and summing the number of minimum-width transistor areas required by each. For details of the transistor level schematics VPR assumes when "building" various FPGA structures, see [6, 7].

To allow averaging of results from circuits of different sizes, we use a normalized area metric: the number of minimum-width transistor areas per tile (i.e. per logic block). All the results in this paper list only the area of an FPGA's routing, since the logic block is held constant throughout all the experiments. The logic block used in this paper occupies 1678 minimum-width transistor areas, and hence the addition of 1678 to any of the routing area results presented in this paper yields the total area per tile.

Prior researchers have evaluated routing area either by counting the number of tracks per channel required to successfully route, or by counting the number of programmable switches in the routing. Counting the number of tracks required to route a circuit is not a good area metric for architecture studies (such as this study) in which the number of switches per track segment can vary, since the area required by each routing track is then variable. Counting the number of programmable switches in the routing is a better area metric, but is still not sufficiently accurate for our purposes. Modern FPGAs use three different types of programmable switch, and the different switches require considerably different layout areas. The connection blocks from routing tracks to logic block input pins are implemented with multiplexers; the connection blocks from logic block output pins to routing tracks, and some of the routing switches, are implemented via pass transistors; and some routing switches are tri-state buffers. Table 1 lists the area required by each of these switch types, including any area required by SRAM bits to control each switch. The area per switch varies by a *factor of 6.8* from the most area-efficient switch to the least area-efficient. Clearly, simply counting the number of programmable switches in a routing architecture does not provide a good area estimate.

Table 1: Comparison of programmable switch areas.

Switch Description	Minimum-Width Transistor Areas
Multiplexer (32 inputs)	2.88 per switch (92 / 32 inputs)
Multiplexer (4 inputs)	4.5 per switch (18 / 4 inputs)
Pass transistor (10x minimum drive)	11.5
Tri-state buffer (5x minimum drive)	19.7

4 Experimental Results: Single Wire Segment Length Architectures

In this section we evaluate architectures in which every routing wire segment has the same length, and in which all the routing switches in switch blocks are tri-state buffers. We ran the twenty largest MCNC benchmarks through the flow of Figure 3 and deter-

1. The Xilinx commercial FPGA router optimizes a more advanced (Penfield-Rubinstein) delay model [31].
2. FPGA architects at both Xilinx and Altera have confirmed to us that transistor area determines the die size of their current FPGAs.

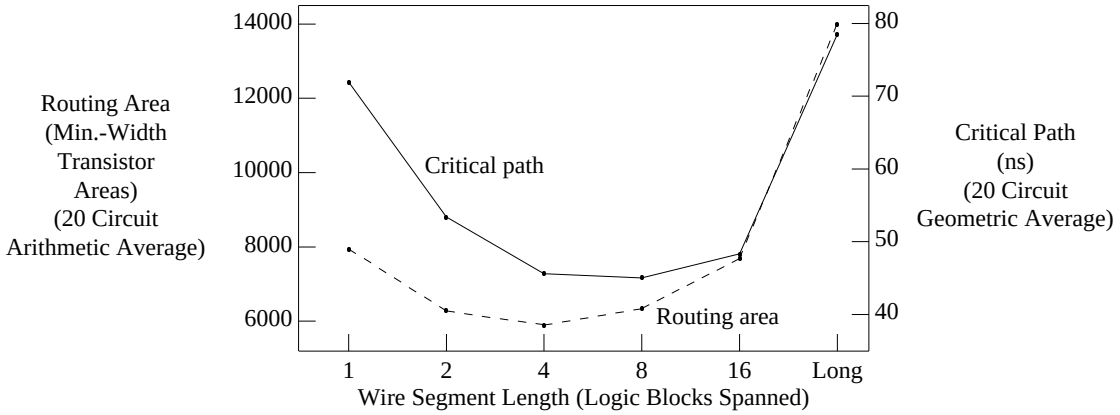


Figure 4: Speed and area of FPGAs vs. routing wire segment length.

mined the routing area required and the critical path delay achieved by each circuit in each architecture of interest. Throughout this paper we compare architectures on the basis of their average performance across the 20 benchmark circuits; the individual circuit results track the overall average quite well, however.

The solid curve in Figure 4 is the average (over the twenty circuits) of the critical path delay for each architecture, while the dashed curve is the average routing area required in each architecture. The horizontal axis in both Figures 4 is the length (in logic blocks) of the routing wire segments; the "Long" point refers to an architecture in which each routing wire spans the entire chip (a long line in Xilinx's terminology). Recall that each architecture in Figure 4 potentially contains a different number of tracks per channel — each circuit is routed in an FPGA with 20% more routing tracks than the minimum the circuit needs to route in an FPGA with the given architecture.

From Figure 4, one can see that the fastest FPGA which uses only one length of wire uses wires of length 4 or length 8. Shorter wires lead to poor speed because long connections must pass through too many buffers. Very long wires degrade speed in two ways. First, short connections are forced to use long wires, which are slower than short wires due to their larger capacitance. Second, even connections that travel the entire length of a long wire become slow when the wire is too long because the metal resistance of the wire becomes large, and eventually reduces speed below that of a larger number of short wires connected by buffers.

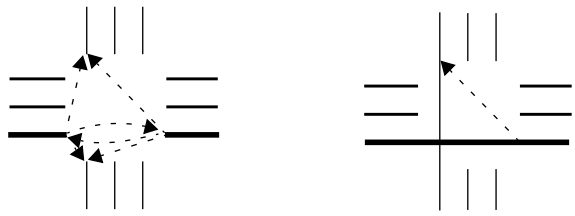
Figure 4 also shows that wire segments of length 4 lead to the most area-efficient FPGA architecture. As we increase the length of the routing wires two competing factors determine the resulting architecture's area-efficiency. First, longer wires are less "flexible"; they cannot be split in the middle, so short connections will waste part of a wire segment. This means the number of tracks per channel required to successfully route a circuit increases as the wire segment length increases. On the other hand, longer wires pass through more switch blocks before terminating, so the fraction of "internal" switch points in switch blocks increases. As Figure 5 shows, these internal switch points require fewer programmable switches, resulting in decreased area. When the disjoint switch block is employed, each internal switch point requires only one programmable switch, while "end" switch points require six programmable switches.

Length 4 wire segments achieve the best combination of low delay and high area-efficiency. An architecture using all length 8 wires can achieve slightly (1.3%) better speed, but requires 7.4% more routing area. While real FPGA architectures can of course use more than one wire length, this result is evocative. It leads one to expect that the best FPGA architectures will either include significant numbers of length 4 or length 8 wires, or will include some wires shorter and some wires longer than length 4 or 8.

4.1 Area Model Revisited

Recall that our area-efficiency metric throughout this paper is transistor area, since current commercial FPGAs are dominated by transistor area. To confirm that the FPGAs we evaluate in this paper are all transistor-area limited, we also monitored the average number of tracks per routing channel (W_{min}) required by each architecture to route the benchmark circuits, since this indicates the amount of routing metal area required by the FPGA. We found that the architectures that were efficient in terms of transistor area generally had average W_{min} values (and hence metal routing area) within a ±20% range, and all these architectures are clearly transistor-area limited. An FPGA employing all length 1 wires, for example, requires 17.6% fewer tracks per channel than an FPGA employing length 4 wires, while a length 8 FPGA requires 20.9% more tracks per channel than a length 4 FPGA.

A few architectures with very poor area-efficiency according to our transistor-based area metric had much greater W_{min} values, however. An FPGA employing all long lines, for example, requires 3.2x as many tracks per channel as an FPGA employing length 4 wires. Depending on the number of metal layers available then, such an architecture may become metal (rather than transistor) limited. If such poor architectures are in fact metal-limited, rather than transistor-limited, they are simply even worse choices than the results presented here indicate.



(a) "End" switch point
6 switches required
by bold wires

(b) "Internal" switch point
1 switch required
by bold wire

Figure 5: Switches required by a disjoint switch block at (a) end and (b) interior of wires.

5 Experimental Results: Two Types of Wire Segment Architectures

In this section we examine somewhat more complex architectures: those that contain two *types* of wire segments. Two wire segments are of different types if their lengths are different, or if they use different types of routing switches to connect to other wires (e.g. pass transistors vs. tri-state buffers).

5.1 Tri-State Buffer Routing Switches Only

We investigated a large number of architectures that contained two different lengths of routing wires, and in which all the switch block routing switches were tri-state buffers [6]. We found that we could achieve only small improvements compared to the best single wire length architecture (length = 4). Table 2 compares the performance of the two best architectures explored in this section to the best single wire length architecture. Both these architectures are fairly similar to an architecture in which all wires have length 4 — one has 25% length 2 wires and 75% length 8 wires, while the other has 75% length 4 wires and 25% length 8 wires. The average speedup vs. a length 4 architecture is only 4.2% for the first architecture, and 4.9% for the second. Both of these architectures are slightly *less* dense than an architecture that contains only length 4 wires. Clearly length 4 wires provide an efficient way to make both short and long connections!

Table 2: Best buffered, two different length architectures vs. best single-length architecture (20 circuit average).

Architecture	Critical Path Delay (ns)	Speedup vs. Length 4, Buffered FPGA	Routing Area (Minimum-Width Transistor Areas)	Routing Area vs. Length 4, Buffered FPGA
All length 4, buffered	45.57	—	5901	—
25% length 2, 75% length 8	43.74	+4.2%	6034	+2.3%
75% length 4, 25% length 8	43.44	+4.9%	5948	+0.8%

5.2 Length 4 Buffered Wires Plus Pass-Transistor-Switched Wires

Since a routing architecture composed solely of length 4 wires that use tri-state buffers as their routing switches performs so well, in this section we investigate architectures in which some routing tracks contain wires of this type. The other routing tracks contain wires that connect to each other with pass transistor routing switches. We will investigate different lengths of these "pass-transistor-switched" wires, and different proportions of the two types of wires.

The solid line in Figure 6 shows the speed achieved by FPGA architectures in which 50% of the routing tracks use length 4 wires connected by buffered switches, and the other 50% consist of some other length of wires connected with pass transistors. The dashed line in Figure 6 shows the routing area required by each architecture. The horizontal axis in Figure 6 is the length of the pass-transistor-switched wires used. The best area-efficiency occurs when the pass-transistor-switched wires are length 2, but length 4 and length 8 wires also have reasonable area-efficiency, and they lead to superior speed.

Figures 7 and 8 investigate the performance of length 4 buffered wires combined with either length 1, 2, 4, or 8 pass-transistor-switched wires in different proportions, (i.e. not just 50 / 50). The horizontal axis in these figures is the fraction of routing tracks composed of the pass-transistor-switched wires; the remainder of the routing tracks are composed of length 4 buffered wires. The "0" point on the horizontal axis corresponds to an architecture composed solely of length 4 buffered wires, while the "1" point corresponds to architectures composed solely of wires that connect to each other via pass transistors. Figure 7 shows the speed of the various architectures, while Figure 8 shows their routing area. Clearly, adding length 1 wires to an architecture is not a good idea. If 33% of the routing tracks are length 1 pass-transistor-switched wires, then area-efficiency improves by 8% (vs. all tracks being length 4 buffered wires), but speed degrades by 7%. Larger fractions of length 1 wires degrade both area and speed — these wires are simply too short to be of much use. Many commercial architectures make heavy use of length 1 wires [3, 4, 33], but our results suggest they could improve both their speed and area-efficiency by using longer wires instead. Note that the most widely studied architecture in academic research, an architecture composed entirely of length 1 wires connected by pass transistors, has extremely poor speed — it is 2.8 times slower than the fastest architecture in Figure 7.

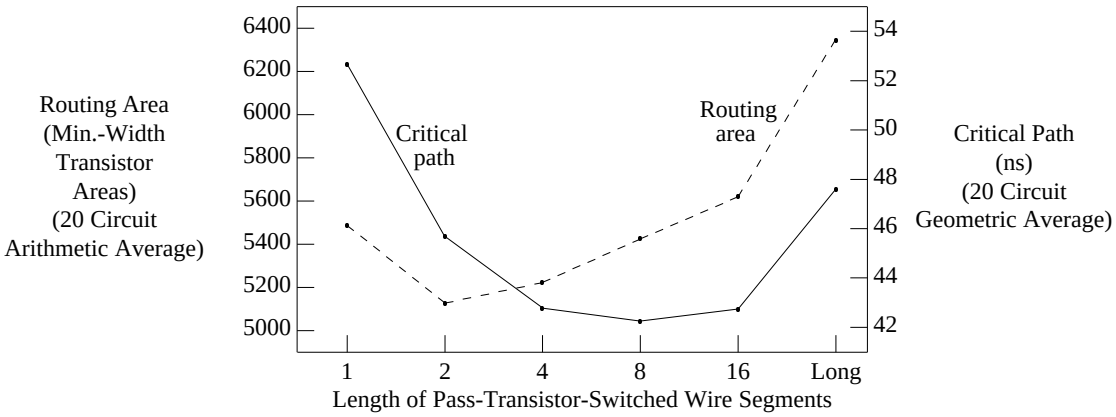
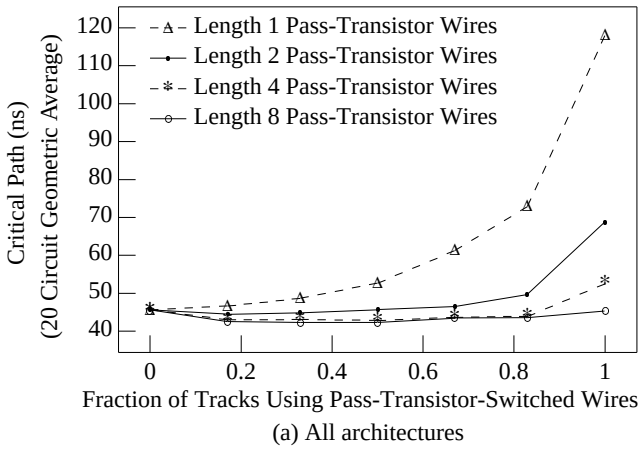
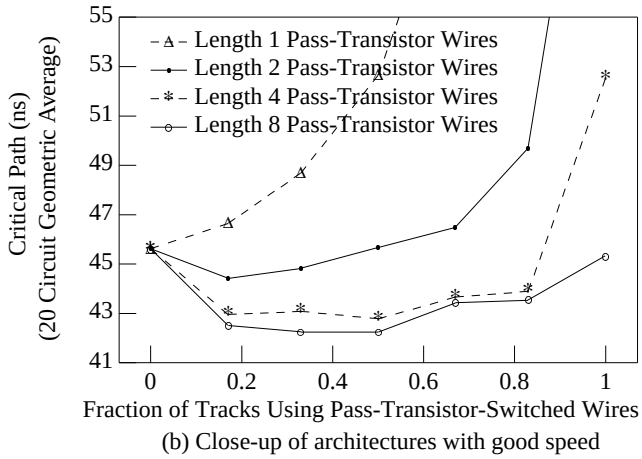


Figure 6: Comparison of FPGAs with 50% length 4 buffered and 50% pass-transistor-switched wires.



(a) All architectures



(b) Close-up of architectures with good speed

Figure 7: Speed of FPGAs with a mix of length 4 buffered wires and pass-transistor-switched wires.

Adding longer pass-transistor-switched wires to an architecture yields better results. Figure 7 shows that making between 17% and 83% of the routing tracks pass-transistor-switched wires of length 4 or 8 increases the FPGA speed. Pass transistors do not have the intrinsic delay of the multi-stage buffers used in buffered routing switches, and they have higher drive strength than a tri-state buffer for the same area, since they use only one transistor, rather than several. On the other hand, the delay through a series chain of pass transistors grows quadratically with the number of pass transistors. The net effect is that pass transistor switches are faster than buffers for connections that pass through a few series switches, but buffers are faster for connections that pass through many series switches. If long wires are used, fewer series routing switches are needed for long connections, making pass transistor switches more competitive with buffers for longer connections. Consequently a mix of moderate length buffered and moderate length pass-transistor wires leads to better speed than using all buffered routing switches or all pass transistor routing switches.

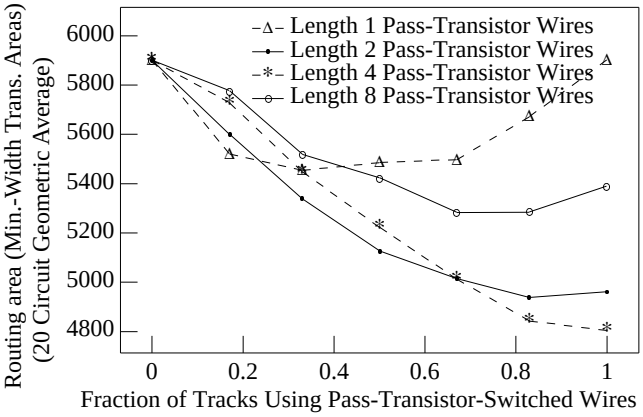


Figure 8: Area of FPGAs with a mix of length 4 buffered wires and pass-transistor-switched wires.

Figure 8 shows that increasing the fraction of routing tracks using length 2, 4 or 8 pass-transistor wires improves the FPGA area-efficiency until this fraction reaches approximately 83%; after that area-efficiency degrades (or levels off, for length 4 wires). A pass transistor switch requires less area than a tri-state buffer, and since pass transistors are bidirectional, one pass transistor can replace two tri-state buffers in the routing. On the other hand, pass transistor switches are not well-suited to routing high-fanout nets. To maintain reasonable speed, a high-fanout net routed using pass-transistor switches tends to use a "star" topology. This requires more wiring, and hence more routing tracks, and hence more area. Making all routing switches pass transistors forces even high-fanout nets to be routed using pass transistors, degrading area-efficiency.

Considering both area and speed, the best architectures use 50% - 83% pass-transistor switched wires of length 4 or 8. Architectures with 50% pass-transistor-switched wires achieve the best speed, while those with 83% pass-transistor-switched wires achieve the best area-efficiency. A major conclusion to be drawn from these results is simply that the best routing architecture contains a mix of pass transistors and tri-state buffers. This fact is not widely known. Prior academic research has focused on FPGAs that contain only pass transistors, while a new FPGA company has made the fact that their FPGAs contain no pass transistors (all routing switches are tri-state buffers) a marketing feature [34].

5.3 Length 8 Buffered Wires Plus Pass-Transistor-Switched Wires

We have found that combining length 8 buffered wires with some pass-transistor-switched wires also results in good FPGA architectures. Using length 8 buffered wires instead of length 4 buffered wires slightly increases the FPGA speed, at the cost of slightly decreased area-efficiency. Otherwise, the architectural conclusions we found when combining pass-transistor-switched wires with length 8 buffered wires are very similar to those of the previous section:

1. The best combination of area and delay results when the pass-transistor switched wires are of length 4 or 8.
2. The best architectures contain from 50% to 83% pass-transistor switched routing tracks, with the 50% pass-transistor architectures giving the best speed, and the 83% pass-transistor architectures yielding the best area-efficiency.

6 Overall Architecture Comparison

In the previous sections we have examined FPGA routing architectures of gradually increasing complexity and looked for important architectural trends by varying the key architectural parameters. In this section we provide an overview by comparing some of the best architectures we have found against each other

and against a routing architecture that is similar to that of the popular Xilinx XC4000X series FPGAs [33, 35]. This "4000X-like" architecture contains 25% length 1 wires, 12.5% length 2 wires, 37.5% length 4 wires, and 25% "one-quarter longs", whose length is one-fourth of the chip. The length 1 and 2 wires connect via pass transistors, while the longer wires connect via tri-state buffers. As well, pass transistor switches also allow the length 4 wires to connect to the length 1 and 2 wires, and the one-quarter longs to connect to length 1 wires. While this routing architecture is very similar to that of the Xilinx XC4000X, it simplifies a few features [35].

Table 3 compares the speed and density of some of the best architectures found in each of the preceding sections to those of this 4000X-like architecture. We also include the performance of an FPGA composed entirely of length 1 wires connected by pass transistors, since most prior FPGA research has focused on this architecture.

All the architectures in Table 3 allow each logic block input pin to connect to 0.5·W routing tracks (i.e. $F_{c,input}$ = 0.5·W) but each logic block output pin can connect to only 0.25·W tracks (i.e. $F_{c,output}$ = 0.25·W). Setting $F_{c,output}$ to 0.25·W instead of the 0.5·W we used in the previous sections reduces the FPGA routing area by 2% to 5%, depending on the exact architecture. Recall that the connection block between the routing tracks and a logic block input pin consists of an $F_{c,input}$ multiplexer, while the connection block from a logic block output pin to the routing tracks consists of $F_{c,output}$ pass transistors controlled by $F_{c,output}$ SRAM bits. Consequently, connections to logic block output pins require more area than connections to input pins (see Table 1), and it is best to make $F_{c,output}$ smaller than $F_{c,input}$.

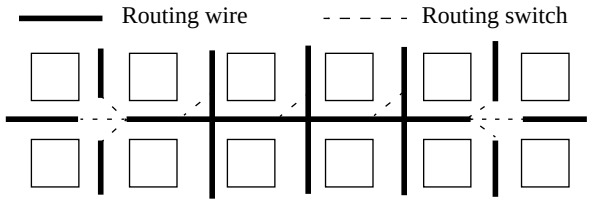
The architectures are listed (after the 4000X-like architecture) in order of increasing complexity. Notice the extremely poor performance of an FPGA using only length 1 wires connected via pass transistors — 150% slower (-60% speedup) and 33% percent larger than the 4000X-like architecture. The best architecture we found using only pass transistors and one length of wire used length 8 wires. This architecture performs much better than a length 1 architecture; it is 5.6% faster than the 4000X-like architecture, at a cost of 16% larger area than that of the 4000X-like architecture. Although the speed and area-efficiency of this length 8, all pass-transistor FPGA are reasonably competitive on average, we consider FPGA architectures that contain no buffers dangerous. As circuit size increases, the longest connections in an FPGA grow longer, and pass through more series switches. Since the delay of pass-transistor switches grows quadratically with the number of switches in series, it is difficult for an architecture that contains only pass transistors to maintain good speed as the size of the logic block array grows. As well, larger circuits can contain nets with a higher maximum fanout, and purely pass-transistor based routing is inefficient for routing high-fanout nets. For both these reasons, architectures that contain only pass transistors do not scale as well with increasing circuit size as architectures that contain some buffers.

The best single-wire-type architecture we found, in which all wires are length 4 and all switches are buffers, is 7.2% *faster* than the Xilinx 4000X-like FPGA. Its area is 30.9% larger, however. It is interesting that such a simple FPGA architecture is reasonably competitive with the complex routing architecture of the 4000X-like FPGA. Simpler routing architectures make it easier to develop CAD tools. As well, they likely make it easier to implement intellectual-property "cores." These cores are sometimes provided as "hard" (placed-and-routed) macros. If there is only one type of routing resource in the FPGA, it is easier to map several of these hard cores into one FPGA and ensure each gets the wires it needs. These factors may make a simple FPGA architecture, in which all the wires are essentially the same, attractive despite its suboptimal speed and area performance.

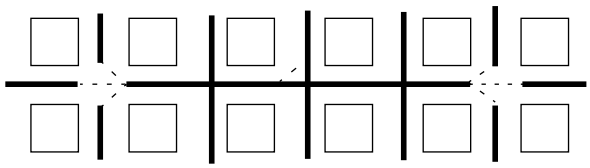
Table 3 also lists four of the best two-wire-type architectures. Each of these architectures combines some buffered wires with some pass-transistor-switched wires. Two of these architectures use only length 4 wires, while the other two use some length 4 and some length 8 wires. Notice that these architectures are all significantly (10.2% to 19%) faster than the 4000X-like architecture, but

Table 3: Comparison of key architectures (20 circuit average).

Segmentation of Routing Tracks, and Switch Types Used	Delay (ns)	Speedup vs. 4000X-like FPGA	Routing Area (Min. Width Transistor Areas)	Routing Area vs. 4000X-like FPGA
Xilinx 4000X-like: 25% L1, 12.5% L2, 37.5% L4, 25% one-quarter longs; mix of buffers and pass transistor switches	48.83	—	4425	—
100% L1, pass-transistor-switched	120.7	-60%	5891	+33.1%
100% L8, pass-transistor-switched	46.22	+5.6%	5131	+16.0%
100% L4, buffer-switched	45.57	+7.2%	5792	+30.9%
67% L4, pass-transistor switched; 33% L4, buffer-switched	42.91	+13.8%	4771	+7.8%
83% L4, pass-transistor-switched 17% L4, buffer-switched	44.31	+10.2%	4569	+3.3%
50% L4, pass-transistor-switched 50% L8, buffer-switched	41.04	+19.0%	5039	+13.9%
83% L4, pass-transistor-switched 17% L8, buffer-switched	43.84	+11.4%	4539	+2.6%
50% L4, pass-transistor-switched; 50% L8, buffer-switched with reduced "switch-block population"	41.23	+18.4%	4708	+6.4%
83% L4, pass-transistor-switched; 17% L8, buffer-switched with reduced "switch-block population"	44.04	+10.9%	4426	0%



(a) Full switch-block population (5/5 or 100%)



(b) Reduced switch-block population (3/5 or 60%)

Figure 9: Examples of different switch-block population values.

none of them is as area-efficient. The area penalty for using an architecture that is 19% faster than the 4000X-like architecture is 13.9%, while the area penalty for an architecture that is 11.4% faster than the 4000X-like architecture is only 2.6%. Both the speed and the area of these architectures are significantly better than the best single-wire-type architecture discussed above, showing that a mix of pass transistors and buffers is very useful in FPGA routing.

The last two lines in Table 3 show the benefits of reducing the *switch-block population* [6, 7] of these two-wire-type architectures so that the buffered wires have a switch block only once every two logic blocks.[1] As Figure 9 (a) shows, ordinarily routing wires can connect to at least one wire segment in every channel they cross. We have found that the area-efficiency of an FPGA can be improved by removing some routing switches from some of the FPGA wire segments, however. Figure 9 (b) shows a routing wire which has programmable switches allowing it to connect to other routing wires only in every second channel (or switch block) it crosses. Reducing the number of programmable switches connecting to some of the wires reduces routing flexibility, and hence more tracks per channel are required for successful routing. The area saved by reducing the average number of switches per wire segment outweighs this cost, however, so the net result is a reduction in routing area. As Table 3 shows, architectures in which the buffered wires can connect to other wires only at every second switch block are from 2.5% to 7% more area-efficient than those that used a switch-block population of 100% for all wires. The exact amount of area improvement depends on the fraction of routing wires that use buffered switches. Notice that one of these architectures is 18.4% faster than the 4000X-like architecture and only 6.4% larger, and another architecture is 10.9% faster and uses the same area as the 4000X-like architecture.

From the results of Table 3 one can see that we have found many architectures with speed superior to that of the 4000X-like architecture, but none with superior density. We believe the 4000X contains too many short wire segments, and this reduces its speed versus many of the architectures we have investigated. The 4000X uses a more advanced switch block than the disjoint switch block used by all the other architectures in Table 3, however. The 4000X switch block contains some switches that allow wires of different lengths to connect, and that allow wires in different tracks to connect. These features make the 4000X switch block more routable than the disjoint switch block, yet it contains only a few more switches than the disjoint switch block. Consequently, this switch block tends to result in FPGAs with superior density. We consider the investigation of better switch block topologies for use with FPGAs that contain some long wires to be a fertile area for future research. We expect that combining the segmentation distributions of some of the architectures listed in Table 3 with a better switch block would lead to significantly improved area-efficiency.

1. Space limitations preclude a thorough discussion of the *switch-block population* issue in this paper. For complete definitions and experimental results showing why it is best to reduce the switch-block population to one switch every two potential locations (rather than one switch every 3 potential locations, for example) see [6, 7].

7 Summary

We have investigated a large number of different routing architecture issues in this work. First, we showed that it is most important for FPGAs to contain wires of moderate length (4 to 8 logic blocks). While most commercial FPGAs contain some very short and some very long wires, we have found FPGAs that use significant numbers of these types of wires to be inferior to those that employ medium-length wires.

We also found that FPGAs that contain a mix of pass-transistor and tri-state buffer routing switches are superior to FPGAs that employ only one type of switch. The fastest FPGAs tend to contain about 50% pass-transistor switches and 50% tri-state buffer switches, while the most area-efficient FPGAs contain about 80% pass-transistor switches and only 20% tri-state buffer switches.

We also found that reducing the switch-block internal population of routing wires that interconnect with tri-state buffers produces an area gain of 2.5% to 7.5% for typical architectures. The switch-block population of these buffered wires should be set so that each wire connects to orthogonal wires at only every second channel it crosses.

Finally, we showed that the best architectures examined in this paper have significantly superior speed to a (slightly simplified) Xilinx XC4000X routing architecture; some architectures are 19% faster than the 4000X architecture. This 4000X-like routing architecture has better area-efficiency than all but one of the architectures we examined, however. The best architectures in this paper have a better area-delay product than the 4000X-like architecture, indicating that they have gained more in speed than they have sacrificed in area. They are also less complex than the 4000X. This is a useful feature, as it simplifies both CAD tool design and the implementation of pre-placed and pre-routed intellectual property cores.

While prior researchers and this work have answered many important questions about FPGA routing architecture, much remains to be done. We believe one of the most fertile areas for future research concerns finding good switch block topologies for use with FPGAs that contain wires longer than length 1. Prior research into this area has focused on switch blocks for use with FPGAs that contain only length 1 wires, but we have found that the best switch block for an architecture with only length 1 wires is not necessarily the best switch block for an FPGA that contains longer wires.

Acknowledgments

The authors are indebted to Jordan Swartz, who modified the FPGA “architecture generator” within VPR to allow targeting of the 4000X-like architecture. We would also like to thank Steve

Young and Steve Trimberger of Xilinx, and Frank Heile of Altera, for helpful discussions on the circuit-level design of FPGAs. This work was supported by the Information Technology Research Centre of Ontario, the Walter C. Sumner Foundation, an NSERC 1967 Scholarship, a V. L. Henderson Research Fellowship and Xilinx. We are also grateful to the Canadian Microelectronics Corporation and TSMC for providing 0.35 µm process information.

References

[1] S. Brown, R. Francis, J. Rose and Z. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

[2] J. Rose and D. Hill, "Architectural and Physical Design Challenges for One-Million Gate FPGAs and Beyond," *ACM Int. Symp. on FPGAs*, 1997, pp. 129 - 132.

[3] Xilinx Inc., The Programmable Logic Data Book, 1994.

[4] Lucent Technologies, FPGA Data Book, 1998.

[5] Vantis Corporation, "VF1 Field Programmable Gate Array," *Preliminary Data Sheet*, 1998.

[6] V. Betz, "Architecture and CAD for Speed and Area Optimization of FPGAs", *Ph.D. Dissertation*, University of Toronto, 1998.

[7] V. Betz, J. Rose and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, To appear in 1999.

[8] J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," *JSSC*, March 1991, pp. 277 - 282.

[9] B. Tseng, J. Rose and S. Brown, "Using Architectural and CAD Interactions to Improve FPGA Routing Architectures," *ACM Workshop on FPGAs*, 1992, pp. 3 - 8.

[10] Y. Chang, D. F. Wong, and C. K. Wong, "Universal Switch Modules for FPGA Design," *ACM Trans. on Design Automation of Electronic Systems*, Jan. 1996, pp. 80 - 101.

[11] S. Wilton, "Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memories," *Ph.D. Dissertation*, University of Toronto, 1997. (Available for download from http://www.ee.ubc.ca/~stevew/publications.html).

[12] J. Greene, V. Roychowdhury, S. Kaptanoglu and A. El Gamal, "Segmented Channel Routing," *DAC*, 1990, pp. 567 - 572.

[13] K. Roy and M. Mehendale, "Optimization of Channel Segmentation for Channelled Architecture FPGAs," *CICC*, 1992, pp. 4.4.1 - 4.4.4.

[14] K. Zhu and D. F. Wong, "On Channel Segmentation for Row-Based FPGAs," *ICCAD*, 1992, pp. 26 - 29.

[15] M. Pedram, B. Nobandegani and B. Preas, "Design and Analysis of Segmented Routing Channels for Row-Based FPGAs," *IEEE Trans. on CAD*, Dec. 1994, pp. 1470 - 1479.

[16] S. Brown, M. Khellah and G. Lemieux, "Segmented Routing for Speed-Performance and Routability in Field-Programmable Gate Arrays," *Journal of VLSI Design*, Vol. 4, No. 4, 1996, pp. 275 - 291.

[17] S. Brown, M. Khellah and Z. Vranesic, "Minimizing FPGA Interconnect Delays," *IEEE Design and Test Magazine*, Winter 1996, pp. 16 - 23.

[18] P. Chow, S. Seo, J. Rose, K. Chung, G. Paez and I. Rahardja, "The Design of an SRAM-Based Field-Programmable Gate Array, Part I: Architecture," *To appear in IEEE Trans. on VLSI*.

[19] V. Betz and J. Rose, "How Much Logic Should Go in an FPGA Logic Block?," *IEEE Design and Test Magazine*, Spring 1998, pp. 10 - 15.

[20] V. Betz and J. Rose, "Effect of the Prefabricated Routing Track Distribution on FPGA Area-Efficiency," *IEEE Trans. on VLSI*, Sept. 1998, pp. 445 - 456.

[21] Altera Inc., Data Book, 1998.

[22] H. Hseih, et al, "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays," *CICC*, 1990, pp. 31.2.1 - 31.27.

[23] Canadian Microelectronics Corporation, "0.35 mm Mixed-Mode Polycide HSPICE Models," *Confidential Process Documentation*, 1997.

[24] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," *Tech. Report*, Microelectronics Center of North Carolina, 1991.

[25] E. M. Sentovich et al, "SIS: A System for Sequential Circuit Analysis," *Tech. Report No. UCB/ERL M92/41*, University of California, Berkeley, 1992.

[26] J. Cong and 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.

[27] V. Betz and J. Rose, "Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input Sharing and Size," *CICC*, 1997, pp. 551 - 554.

[28] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," *Int. Workshop on Field-Programmable Logic and Applications*, 1997, pp. 213 - 222.

[29] 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.

[30] W. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, Jan. 1948, pp. 55 - 63.

[31] S. Trimberger, Ed., *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, 1994.

[32] R. Hitchcock, G. Smith and D. Cheng, "Timing Analysis of Computer-Hardware," *IBM Journal of Research and Development*, Jan. 1983, pp. 100 - 105.

[33] Xilinx Inc., "XC4000E and XC4000X Series Field-Programmable Gate Arrays," *Data Sheet*, 1997.

[34] P. Clarke, "Dynachip Claims Speed Breakthrough in its FPGAs," *Electronic Engineering Times*, June 9, 1997, p. 10.

[35] J. Swartz, "A High-Speed Timing-Aware Router for FPGAs," *M.A.Sc. Thesis*, University of Toronto, 1998.