Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect

Vaughn Betz and Jonathan Rose

Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ontario, CANADA M5S 3G4 {vaughn, jayar}@eecg.utoronto.ca

Abstract

This paper examines the electrical design of FPGA interconnect circuitry. We explore the circuit design of pass transistor and tri-state buffer routing switches, determine which transistor sizing, metal width and metal spacing are best for FPGA interconnect, and show that FPGA interconnect should be *electrically heterogeneous* -- some (~20%) of the routing tracks should be designed for maximum speed while the remainder should be more area-efficient.

1. Introduction

While considerable research has investigated the optimization of FPGA routing architecture (the lengths of the routing wire segments and the pattern of routing switches used to interconnect them [1]), relatively little has been published concerning the *electrical* optimization of FPGA interconnect. Little is known about the best circuit design and transistor sizing for the routing switches themselves. Similarly, there has been no published work examining the layout of FPGA routing wire segments, despite the fact that using the proper metal width and spacing in deep-submicron processes is crucial to obtain the best circuit speed [2]. The programmable routing accounts for most of the area and most of the delay in FPGAs [1], so fast and area-efficient circuitry is essential.

In this paper we explore four related issues: the circuit design of pass transistor and tri-state buffer routing switches, the best transistor sizes to use in both types of switch, how routing wires should be laid out (what metal width and spacing is best?), and *electrically-heterogeneous* FPGAs, in which some routing wires are tuned for density and some for speed.

Considering the importance of the electrical design of its routing to an FPGA's speed and density, there is relatively little published prior work. In [3, 4], Chow et al discussed the implementation of an SRAM-based FPGA in a 1.2 μm CMOS process, and highlighted many circuit design issues. In [5], Khellah, Brown and Vranesic performed some transistor-sizing experiments on pass transistor routing switches in a 0.8 μm process. In [6], Dobbelaere, Horowitz and El Gamal proposed an innovative regenerative feedback circuit element to speed FPGA routing.

This paper is organized as follows. The next section describes the class of FPGAs we are investigating, and discusses two important circuit issues in the design of FPGA routing switches. In Section 3 we determine which transistor sizes lead to FPGA interconnect with the best area-delay product, and in Section 4 we investigate the effect of different routing wire metal widths and spacings. Section 5 examines "electrically-heterogeneous" FPGAs in which the spacing between some routing wires is wider than others.

This work was supported by the Information Technology Centre of Ontario, the Walter C. Sumner Foundation, an NSERC 1967 Scholarship, a V. L. Henderson Fellowship and the Canadian Microelectronics Corp. The authors are indebted to Steve Trimberger, Steve Young and Bill Carter of Xilinx and Frank Heile of Altera for helpful discussions and to TSMC for process data.

2. FPGA Architecture and Circuit Design

We investigate SRAM-based, island-style FPGA architectures [1]; this is the style of FPGA employed by Xilinx, Lucent Technologies and the Vantis VF1. Fig. 1 shows the key circuitry in such an FPGA's interconnect. Routing switches are either pass transistors or pairs of tri-state buffers (one in each direction), and allow routing wire segments to be joined to form longer connections. Multiplexers allow routing wires to be connected to the input pins of logic blocks, while demultiplexers (a set of pass transistors) allow routing wires to be driven by logic block output pins.

In order to investigate routing transistor sizing, we must first choose values for several topological aspects of the FPGA interconnect. We set these topological parameters to values that were shown to be good choices in [7, 8]. As Fig. 1 shows, each routing wire can connect to three other wires (via three routing switches) at each of its endpoints, and can connect to one wire at each internal point where it crosses an orthogonal channel. Each routing wire can be driven by one output pin at each logic block it spans. A key topological parameter of FPGA interconnect is the *logical length*, or number of logic blocks spanned, by a routing wire. In Fig. 1, for example, the routing wire shown is of length 4. In this work we will investigate appropriate transistor sizings for wires of many different logical lengths.

In order to evaluate the speed of FPGA routing, we must know the *physical length* of a routing wire that spans L logic blocks. Throughout this work we assume each basic tile (a logic block plus it's associated routing) is 300 μm long, so a wire that spans L logic blocks is L·300 μm long. The basic tile for an FPGA architecture with good performance was shown to be 300 μm long (in a 0.35 μm process) in [7, 8], and the length of a Xilinx 4000X layout tile is 340 μm (also in a 0.35 μm process), so this value is reasonable.

2.1. Leakage Current and Gate Boosting

Fig. 2a shows a potential problem with the use of pass transistors in FPGA interconnect. A pass transistor's output voltage only rises to V_{dd} - V_t, where V_t is the transistor threshold voltage (including the increase in the nominal V_t due to the body effect [9]). In TSMC's 3.3V, 0.35 μm CMOS process, for

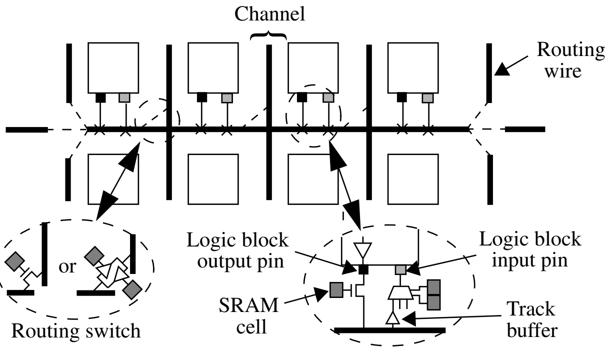

Fig. 1. FPGA routing circuitry.

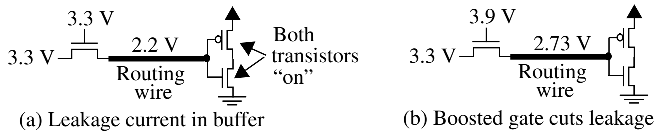

Fig. 2. Gate boosting to solve leakage current problem.

example, the output voltage of the pass transistor will rise only to 2.2 V. This voltage is low enough that downstream buffers (such as the "track buffer" in Fig. 1) which sense the state of the routing wire will have both their pull-down and pull-up transistors partially on. This results in a large leakage current and unacceptable dc power dissipation -- a typical 100 000 gate (8000 4-input look-up tables) FPGA would dissipate 21 W of static power due to this buffer leakage current.

Fig. 2b shows one solution. By boosting the gate of routing pass transistors to 3.9V (one nominal threshold voltage above V_{dd}), we increase the pass transistor output voltage to 2.73V, which is enough to turn off the pMOS pull-up transistor in any downstream buffer. This reduces the static power dissipation of a 100 000 gate FPGA to a much more reasonable 0.041 W. This "gate-boosting" technique has been used by Xilinx in their FPGAs, and we use it throughout this paper.

2.2. Tri-state Buffers

Fig. 3 illustrates two possible tri-state buffer circuits. Circuit (a) has one disadvantage: the buffer drive strength is reduced (for a given transistor size) by the pass transistor on the output. Tri-state buffers built using method (b), however, have several disadvantages. They have higher intrinsic delay, require more area for moderate size buffers, and add more capacitive loading (both from their input and their output) to the routing wires. As well, at each end of an FPGA routing wire there are three routing switches which all have the same routing wire as their input. As Fig. 3c shows, if we implement these switches using method (a), we can build just one inverter chain, rather than 3, saving considerable area. For these reasons, we have found that method (a) is generally the superior technique for building tri-state buffers in FPGA routing, and we use this technique throughout this work.

3. Transistor Sizing of Routing Switches

The metal capacitance of FPGA routing wire segments is quite large in deep submicron processes, so one can significantly increase FPGA speed by increasing the size (and hence the drive strength) of the routing switches. However, since most of the area in an FPGA is due to routing switches [1], the cost in area-efficiency of increasing routing transistor sizes cannot be ignored. We believe the best transistor sizing minimizes the area-delay product of the resulting FPGA, because:

1. Intuitively, we want to increase the routing transistor size until the incremental speed gained by further size increases is not worth the area penalty. Minimizing the area-delay product makes this intuitive goal quantitative.
2. For parallel computations, an FPGA's throughput is: (# of functional units) x (functional unit speed). Phrased another way, the throughput is proportional to (1/area per functional unit) x (1/delay). Hence minimizing the area-delay product of an FPGA maximizes its throughput on parallel algorithms.

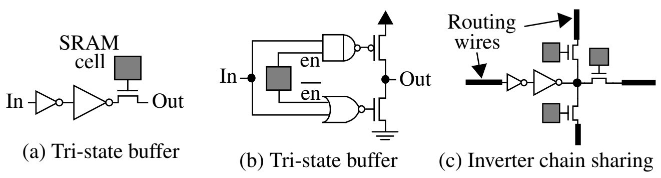

Fig. 3. Methods of building tri-state buffers.

To find the FPGA with the minimum area-delay product achievable with a given logic block and routing architecture, we would have to simultaneously vary the size of every transistor in a basic tile of the FPGA, and determine the area and delay achieved by the FPGA under each transistor sizing. Clearly this is a dauntingly large and complex search space. We can, however, *approximately* minimize the area-delay product of an FPGA by minimizing the routing switch area - wire delay product for each type of routing wire in the FPGA. This is the approach we take to routing transistor sizing.

In the following sections, all delay values are taken from simulations of TSMC's 0.35 µm, three-layer metal CMOS process. We assume all wiring is laid out in minimum width, minimum spacing metal 3 in this section (we examine different metal widths and spacings in Section 4). We estimate the layout area required by the routing switches as a function of the number and sizes of the transistors required to build them [7, 8], including the area of any controlling SRAM bits. Note that it is the transistor area, not metal area, which determines the die size of current commercial FPGAs, so our area model is based on transistor area.[1]

Recent research [10, 7, 8] has shown that FPGA interconnect should contain a mix of wires which connect via pass transistors and wires which connect via tri-state buffers. We investigate sizing issues for both types of routing switch.

3.1. Sizing Pass Transistor Routing Switches

The delay through a chain of N routing wires connected by pass transistors grows essentially quadratically with N [9]. In other words, $T_d \cong D_{dom} \cdot N^2$. We call D_{dom} the dominant delay constant, and we wish to minimize D_{dom} to maximize the speed of pass-transistor-based routing.

Fig. 4 plots this dominant delay constant versus the pass transistor width for routing wires of different logical lengths. The delay constant has been divided by the wire length, L_{wire}, to allow all the curves to be shown on the same scale. The horizontal axis in Fig. 4 is the width, relative to the minimum

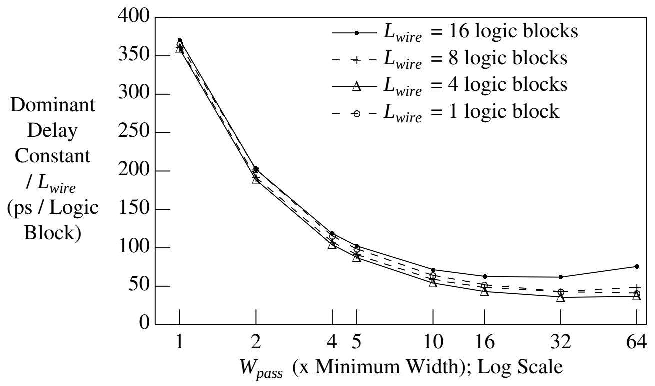

Fig. 4. Dominant delay constant vs. routing pass transistor width.

1. FPGA architects at both Xilinx and Altera have confirmed to us that transistor area determines the die size of their FPGAs.

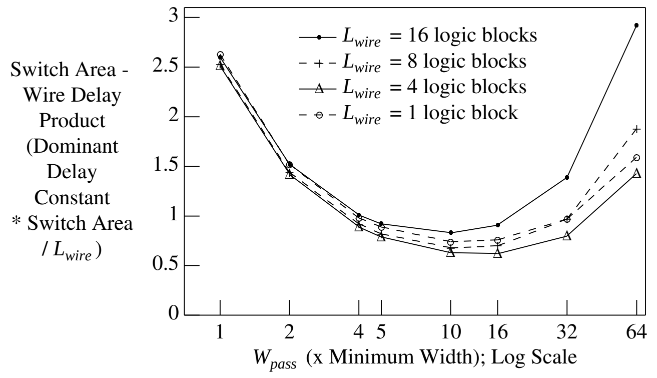

Fig. 5. Switch area - wire delay product vs. routing pass transistor width.

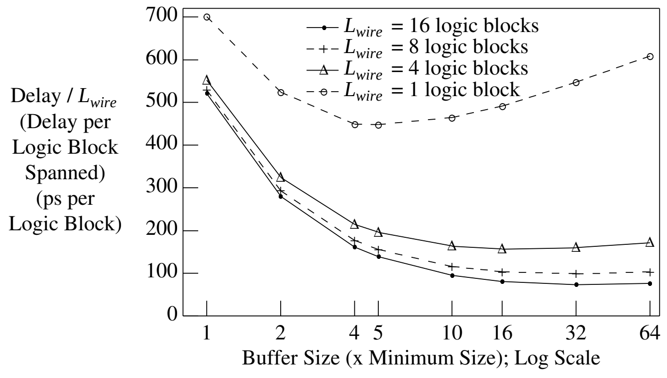

Fig. 6. Delay per logic block spanned vs. routing tri-state buffer size.

contacted transistor width of 0.7 µm, of the routing pass transistors. As the width of the pass transistors increases, the wiring delay clearly drops significantly. For very large pass transistors the delay stops improving, since the switch capacitances (as Fig. 1 shows, there are many routing switches loading each wire) become larger than the metal capacitance and the wire becomes self-loaded. For the longer wires, delay degrades once the transistor width is more than about 30 times minimum, since the metal resistance becomes more significant than the pass transistor equivalent resistance.

Fig. 5 plots the area of a routing switch multiplied by the dominant wire delay constant as a function of pass transistor width. Again, the curves are normalized by dividing by the wire length so they can all be plotted on the same scale. For wire lengths of 1, 4, or 8 logic blocks, transistor widths of 10 and 16 times the minimum are essentially tied for the best area - delay product. The higher end-to-end metal resistance of a length 16 wire, however, makes a pass transistor width of 10 preferable to a width of 16.

3.2. Sizing Tri-state Buffer Routing Switches

To determine the best size for tri-state buffer routing switches, we simulated the delay to pass through a routing buffer and the wire it drives. As in the previous section, we perform this analysis for various routing wire lengths, and we use our area model to assess the area cost of different size buffers. We build a buffer of minimum size with minimum contactable width (0.7 µm) nMOS transistors, while the pMOS pull-up is 1.9 times this width to achieve equal rise and fall times. The larger buffers are multi-stage buffers, and the stage ratio is kept as close to 4 as possible to yield good speed with a small area [9]. We define the size of a buffer as the ratio of the size of the transistors in its final stage to those in the minimum size buffer defined above.

Fig. 6 shows the wire delay divided by the wire length (i.e. the delay to pass one logic block) versus the size of the tri-state routing buffers. As buffer size increases from the minimum size, speeds improve for all wire lengths. Once the buffer is larger than four times the minimum, however, the speed of length 1 wires starts to degrade. This occurs because the increase in buffer intrinsic delay as the buffer grows is larger than the decrease in the time it takes the larger buffer to discharge the routing wire (and attached switch) capacitance. Longer wires continue to see some speed improvement until the buffer size reaches 16 times the minimum (for length 4 wires) or 32 times the minimum (for length 8 and 16 wires).

Fig. 7 shows the wire delay - routing switch area product curve for four different wire lengths. For wires of length 4, 8 and 16 the best switch area - wire delay product occurs with a buffer that is five times the minimum size. For a wire of length 1, the best area-delay product occurs with a buffer size of only twice the minimum.

Notice that the best routing pass transistor width was ten times the minimum, while the best tri-state buffer size is only five times the minimum. There are two reasons for this behaviour. First, as the size of a tri-state buffer is increased, more stages are added to the buffer chain. Thus some of the speed gained by the buffer's increased drive strength is offset by its increased intrinsic delay. Second, since a tri-state buffer contains several transistors, it consumes more area at a given size than a pass transistor. Consequently, as a tri-state buffer is sized up, it more rapidly swamps the fixed area overhead of its controlling SRAM bit, so its area growth is closer to linear in the buffer size than that of a pass transistor.

Although we assumed minimum spacing metal 3 wiring in Sections 3.1 and 3.2, laying out wires in metal 2 or using a different metal spacing or width does not significantly change the point at which the best switch area - wire delay product occurs.

4. Routing Wire Layout

Section 3 assumed that routing wires used the minimum metal width and spacing. Increasing the spacing between metal wires reduces the metal capacitance, while increasing the metal width reduces the metal resistance, at the cost of some increase in the metal capacitance. Of course, increasing either the metal width or the metal spacing increases the metal pitch; this may cause in an increase in the FPGA area (if the metal area

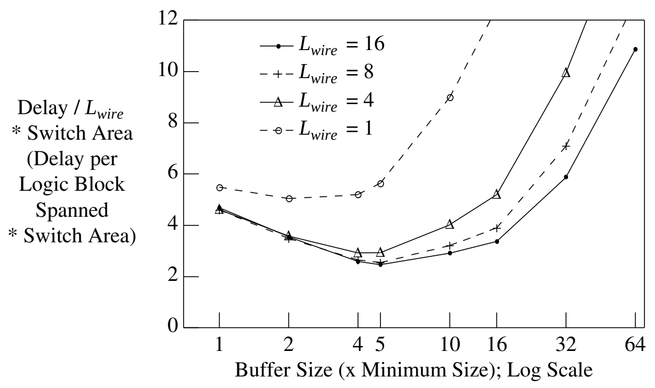

Fig. 7. Switch area - wire delay product vs. routing tri-state buffer size.

becomes too large to fit over the transistor area). In this section we determine if FPGA routing wires benefit from greater than minimum metal width or spacing. We again assume that routing wires are laid out in metal 3.

Table 1 lists the speed benefit from either widening routing wires or increasing the spacing between wires to the point that the metal pitch (width + spacing) becomes 1.55x the minimum. The upper half of the table lists data when the switches between wires are pass transistors, while the bottom half lists data for the tri-state buffer case. We have intentionally presented results for a relatively long (length 16) wire, since such wires have the greatest potential for speed improvements through metal widening. The routing transistors are sized for the best area-delay product in each case.

Table 1: Effect of metal width and spacing on speed of a length 16 wire.

Switch Type	Metal Width (μm)	Metal Space (μm)	Relative Delay
Pass Transistor	0.6 (min)	0.5 (min)	1
	1.2 (2x min)	0.5 (min)	0.86
	0.6 (min)	1.1 (2.2x min)	0.75
Tri-state Buffer	0.6 (min)	0.5 (min)	1
	1.2 (2x min)	0.5 (min)	1.03
	0.6 (min)	1.1 (2.2x min)	0.76

Table 1 shows that increasing metal spacing yields greater speed gains than widening metal. In fact, for buffered routing switches, widening the wire leads to a 3% delay increase, even for this fairly long, length 16, wire. Since buffers prevent the "build-up" of metal resistance when several routing wires and switches are connected in series, widening wires connected by buffers is not very effective -- the increase in wire capacitance outweighs the benefits of decreased wire resistance. Widening wires performs even more poorly for shorter routing wires.

It may seem surprising that widening the routing wires is so ineffective, since standard cell designs often use wider than minimum metal traces. In standard cell designs, one can increase the size of the buffer driving a wire to compensate for the increase in wire capacitance due to wider metal. In FPGA routing, however, we have not one buffer or pass transistor driving a wire, but many possible drivers hanging off the wire along its length. The area cost of sizing up all these switches to compensate for the increased metal capacitance is significant. Normally, therefore, these switches are small enough that their equivalent resistance is higher than the metal resistance of even a fairly long, minimum width wire.

5. Electrically Heterogeneous Routing

We have found that it is best to optimize some of an FPGA's routing for speed, while the remainder is optimized for density. For example, Fig. 8 shows the average speed achieved by a set of 20 benchmark circuits when implemented in a realistic FPGA architecture by the VPR timing-driven routing tool [7, 8]. The vertical axis is the critical path delay while the horizontal axis is the fraction of routing tracks per channel laid out width a wide (2.5 μm) metal spacing. The remaining routing tracks are laid out with the minimum (0.5 μm) metal spacing. Notice that an FPGA in which all the wires are widely spaced out is 15% faster than an FPGA in which all wires use the minimum spacing. Even more interesting is the fact that by spacing out only 20% of the tracks we obtain a 13% circuit speedup. Increasing the spacing of only 20% of the routing tracks mini-

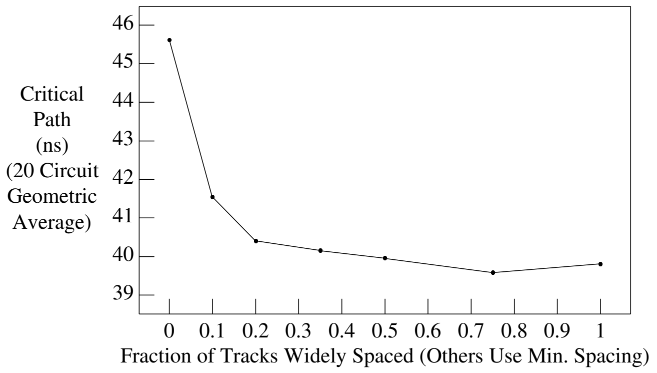

Fig. 8. Speed of a realistic FPGA vs. routing wire spacing.

mizes the increase in metal area required, but still yields almost all of the achievable speedup. Note that a good timing-driven router [7] is key to realize these speed gains -- the router must correctly identify the critical connections and route them on the extra-fast wires.

Instead of (or in addition to) spacing out some routing tracks, one could make the switches attached to the wires in certain tracks extra large. From the results of this section, we can predict that sizing up the switches on only 20% of the tracks will produce almost as much speed gain as sizing up every switch.

6. Conclusions

This paper demonstrated a solution to the leakage current problem caused by pass transistors, and showed which form of tri-state buffer is best suited to FPGA interconnect. We found that in a 0.35 μm process it is best for routing pass transistors to be approximately ten times the minimum contactable width, and for tri-state buffers to be five times the minimum size. We also found that widening FPGA routing wires does not improve their speed as much as increasing their spacing. Finally, we showed that it is best to make FPGA interconnect electrically heterogeneous -- some (~20%) of the routing tracks should be designed for speed, with the others designed for area-efficiency.

References

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

[2] J. Cong, L. He, C. Koh and Z. Pan, "Global Interconnect Sizing and Spacing with Consideration of Coupling Capacitance," *ICCAD*, 1997, pp. 628 - 633.

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

[4] P. Chow, S. Seo, K. Chung, G. Paez, and J. Rose, "A High-Speed FPGA Using Programmable Mini-Tiles," *Symp. on Integrated Systems (formerly Conf. on Advanced Research in VLSI)*, 1993, pp. 104-122.

[5] M. Khellah, S. Brown and Z. Vranesic, "Modelling Routing Delays in SRAM-Based FPGAs," *Cdn. Con. on VLSI*, 1993, pp. 6.B.13 - 6.B.18.

[6] I. Dobbelaere, M. Horowitz and A. El Gamal, "Regenerative Feedback Repeaters for Programmable Interconnections," *ISSCC*, 1995, pp. 116 - 117.

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

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

[9] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, Second Edition*, Addison-Wesley, 1993.

[10] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density," *To appear in FPGA 1999.*