Directional and Single-Driver Wires in FPGA Interconnect

Guy Lemieux Edmund Lee Marvin Tom Anthony Yu
Department of ECE, University of British Columbia
Vancouver, BC, Canada

E-mail: { lemieux | eddyl | marvint | anthonyy } @ ece.ubc.ca

Abstract

Modern FPGA architectures from Altera and Xilinx have shifted away from allowing multiple drivers to connect to each interconnect wire. This paper advocates the need for this shift to single-driver wiring by investigating the necessary architectural and circuit design changes. When single-driver wiring is used, area improves by 25%, delay improves by 9%, and area-delay improves by 32% compared to bidirectional wiring. Wiring capacitance is reduced by 37% due to reduced switch loading and physical wire length shrinkage. Furthermore, it is shown that **larger** *circuits tend to realize* **larger** *savings. No significant CAD tool changes are needed.*

1. Introduction

To support larger logic capacities, FPGAs must be built with more logic elements and larger interconnection networks. The interconnection network typically dominates in all key metrics: area, delay, and power. To extract every bit of performance, it is necessary to consider both implementation details and architectural efficiency. In this paper, we consider two circuit-oriented optimizations that will impact FPGA architecture and improve both area and delay.

The first optimization is the policy of creating *directional wires*. Conventional bidirectional wires are connected with bidirectional switches, *e.g.* two back-to-back tristate drivers. However, once configured, an FPGA always uses the switch in only one direction. This leaves at least 50% of all tristate drivers un-utilized. With directional wires, drivers are needed in only one direction. They will be more highly utilized if the number of wires in each direction closely matches the number of nets travelling in the same direction. This work shows that unmodified CAD tools can automatically achieve high utilization in both directions.

The second optimization considered in this paper is the strict use of *single-driver wiring*, where there is only one driver for every interconnect wire. This means that tristate drivers are replaced with regular (non-tristate) drivers. This can reduce area overhead (the transistors implementing the tristate ability are removed) and improve drive strength (for a fixed driver size). However, to achieve routing flexibility, these drivers must have some type of selection ability on the input, *e.g.*, using a multiplexer. This multiplexer selects from all possible sources, including both connections inside the switch block as well as the CLB outputs. Since this multiplexer will have many inputs, it introduces delay overhead. This work shows that single-driver wiring results in net improvements to both area and delay.

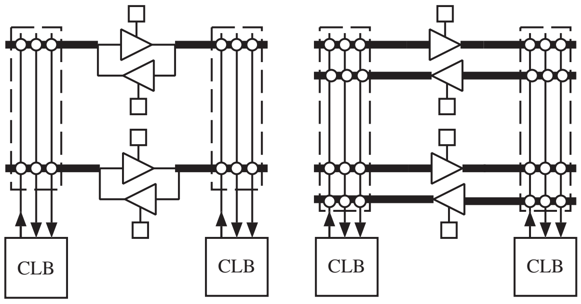

Figure 1. Bidirectional and directional wires.

A comparison of bidirectional wires and directional wires is shown in Figure 1. Notice the maximum-possible number of wires in each direction is the same. However, for the same total number of drivers, directional wiring provides *twice as many total wires* as bidirectional wiring. Unfortunately, this also increases the amount of area needed for connections to/from the configurable logic blocks (CLBs). To save area, there *must* be fewer than twice as many directional wires as bidirectional wires. The amount that can be reduced depends upon what fraction of signals are flowing in each direction. This work shows a large net area savings because directional and bidirectional wiring needs approximately the same number of tracks.

The use of single-driver wiring also requires two important changes to the detailed routing architecture. First, CLB outputs can only be driven onto wires

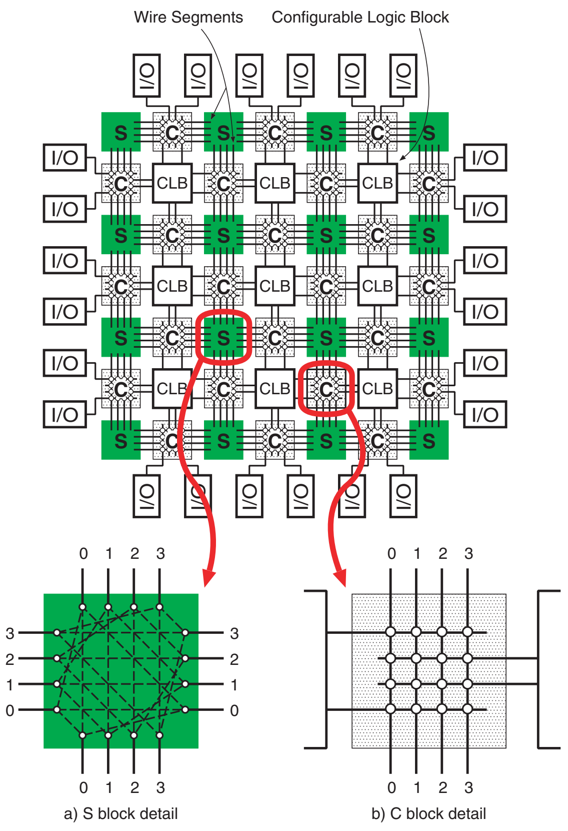

Figure 2. Island-style FPGA architecture.

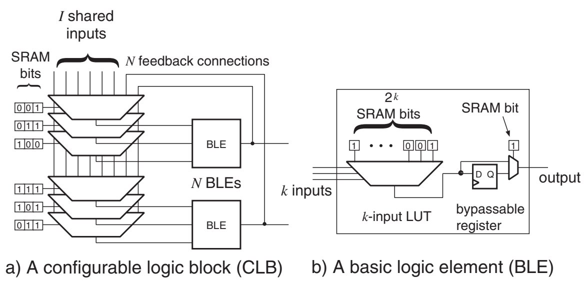

Figure 3. CLB and BLE logic structure.

that begin nearby (where the driver is located). Of course, CLB inputs can be received from any point along the *midpoints* of the wire or at the very end. Second, switch block turns from the midpoint of a wire to the midpoint of another wire are no longer possible. Instead, turns from midpoints can be made only to the beginning of nearby wires. Hence, single-driver wiring places a new restriction on routing connectivity. This paper determines that these new restrictions are as routable as previous architectures.

2. Architecture

The general architecture considered in this paper is depicted in Figures 2 and 3. A k-input lookup table and flip-flop are combined into a *basic logic element*, or BLE. A total of N BLEs are grouped together into a larger logic unit known as a *configurable logic block* or CLB. In this work, $k = 4$ and $N = 6$ is used. Each CLB shares $I = 14$ inputs among the BLEs. Previous work has shown this architecture is near-minimum in both area and area-delay product [1, 5]. Since FPGAs tend to be interconnect-dominated, the results in this paper would be similar for any other reasonable choices of k, N, or I.

The interconnect organization in Figure 2 is called an island-style or mesh architecture. In this style, CLBs are placed into a regular 2-dimensional grid. Between the rows and columns of CLBs are horizontal and vertical routing channels. The portion of a channel adjacent to one CLB is a *channel segment*. The *channel width*, W, is the number of wiring tracks, numbered 0 through $W-1$, in each channel segment. In this work, we assume wires are of *logical length* $L = 4$, meaning they span 4 channel segments.

A *switch block* or S block is formed everywhere the horizontal and vertical channels intersect. S blocks contain switches which are used to connect two interconnect wires. This allows nets to turn corners or extend farther down a channel. The S block in Figure 2a uses a dashed line to indicate a possible connection (*i.e.*, a switch) between wires.

A *connection block* or C block is formed where CLB input or output pins are connected to the routing channel. Each CLB pin is distributed and appears only once among the four sides. Figure 2b shows four pins connected to every track. We assume each input pin is connected to 50% of the tracks. Output pin connectivity varies and will be discussed later.

The VPR place-and-route toolset [3] is used to map benchmark circuits into the above architecture. Local modifications made to support more architectural features and improved delay calculations are described in [7]. VPR has also been further modified to reflect the changes to the routing architecture and circuit design details presented herein. However, the place and route algorithms have not been changed.

3. Implementation Details

3.1. Directional Wires: Switch Blocks

The connections in a switch block for wires of length 1 are shown in Figure 4. The bidirectional switch block, shown on the left, is implemented using buffers and multiplexers in a manner similar to other work [3, 8]. This circuit is functionally equivalent to the Xilinx XC4000-series PIP [10] which employs pass transistors. The buffer is formed with tapered inverters, and a wide NMOS pass transistor on the buffer output creates the tristate output. Each tristate buffer drives only one wire and a multiplexer is used to choose one of neighbouring wires as the input [6, 9]. The multiplexer transistors are small, but a small buffer (inverter) is used to isolate them from the

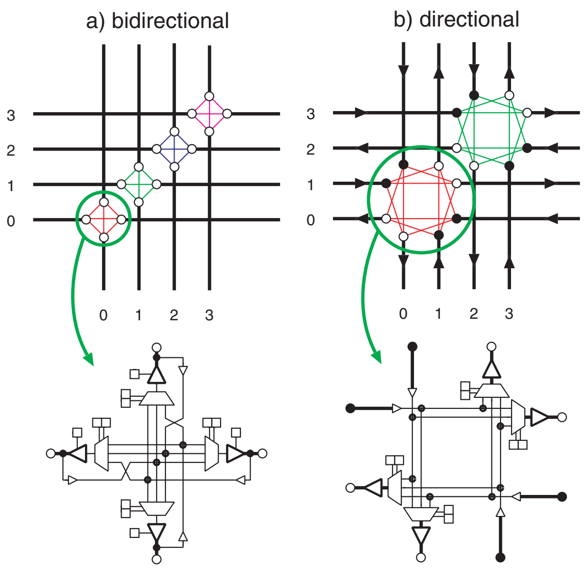

Figure 4. S blocks, length 1 wires.

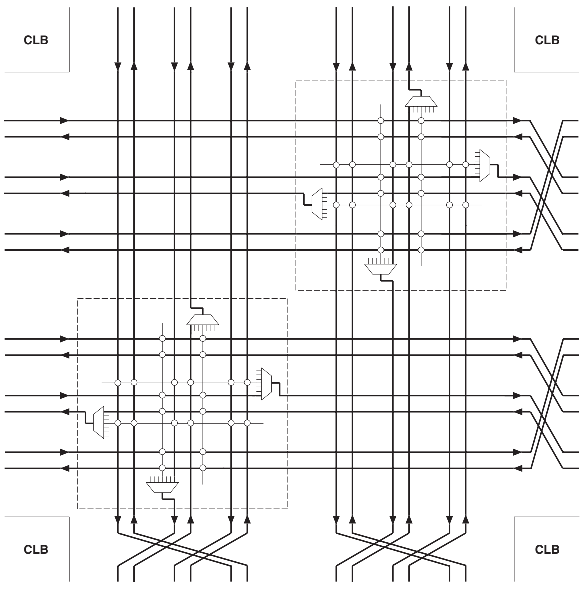

Figure 5. Directional S block, length 3 wires.

wires they select. SRAM elements control the select inputs of the multiplexer and the tristate.

In contrast, a directional switch block is shown on the right in Figure 4. Here, a convention is adopted so that signals will flow down/left on even-numbered tracks and up/right on odd-numbered tracks. As a result, signals may change to a different track number when turning. The number and type of circuit components to implement a directional switch block element are identical to a bidirectional element. However, they are arranged slightly differently to connect two horizontal and two vertical wiring tracks. As well, the tristate buffers can be replaced with regular drivers if single-driver wiring is employed. Notice also that the circuit for directional wiring requires an even number of tracks.

To build an island-style FPGA with long wires (delay requirement) using one layout tile (practical requirement), it is necessary to add track rotations to the wires. The rotations needed for directional wires of length $L = 3$ are shown in Figure 5. This produces a naturally staggered starting pattern for the wires.

Figure 5 also illustrates two other important details. First, notice that long wires produce exactly L times as many wiring tracks as Figure 4. Recall that directional wiring uses a switch element with pairs of wires. Hence, to be practical, **the channel width must be a multiple of $2L$ for directional wiring, or a multiple of L for bidirectional wiring**. VPR was modified to enforce this quantization in the architecture. Previously, VPR's architecture generator assumed that the channel width could be any integral value, resulting in some architectures which were impractical to layout.

Second, the muxes inside the switch elements of Figure 5 illustrate additional connections that can be made at wire midpoints. For clarity, numerous mux inputs are represented by a single wire and the possible connections to choose from are drawn using open circles. Additional mux inputs that are shown as unused in this figure represent connections available for CLB output pins (discussed below). We chose to build a reasonably rich interconnect structure, so every wire midpoint that passes through a switch block is connected to the two muxes corresponding to the two possible turns it can make. It may be possible to reduce (or even increase) this amount of connectivity, but that is not investigated here.

3.2. Single-Driver Wires: Connection Blocks

With bidirectional wiring, CLB output pins can connect to any track. Figure 6 shows how one output pin can efficiently connect to multiple tracks. This design uses a shared-buffer to drive multiple wide pass transistors. Each pass transistor forms a tristate output that can be independently turned off. When multiple transistors are on, delay is adversely affected. In this work, we assume each of the N CLB output pins connect with $1/N$ of all tracks.

With directional wiring, it is still possible to connect CLB output pins to any track by employing tristate buffers as shown in Figure 6. We call this a *directional-tristate* (dir-tri) architecture. For this architecture, we assume each CLB output connects to $1/N$ of all tracks.

To employ single-driver wiring, tristate elements cannot be used. Instead, CLB outputs must connect to the input muxes of wires that begin nearby. Hence, the type of connections shown in Figure 7 are the only ones allowed. This places a restriction upon the interconnect architecture which has not been previously

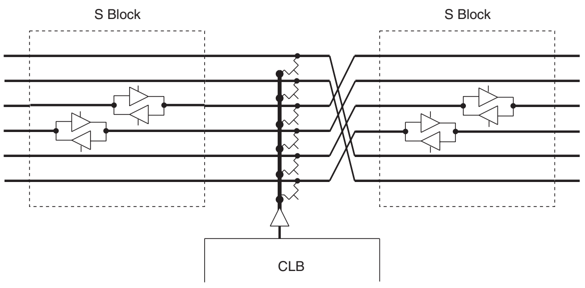

Figure 6. CLB output with tristate drivers.

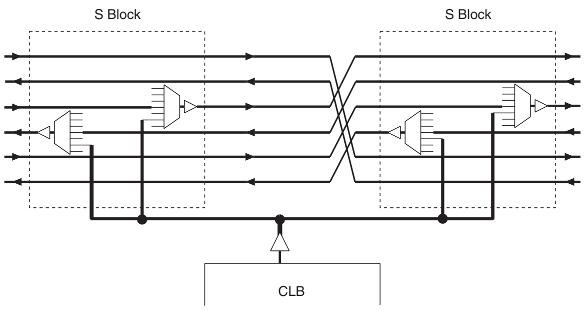

Figure 7. CLB output for single-driver wiring.

considered. We wish to build a reasonably rich interconnect, so we connect an output pin to **all** of the wires lying in the same channel that begin in the two adjacent switch blocks. This means that each output pin connects to $2/L$ of the wires in a channel. It may be possible to reduce this for additional area savings, but we do not consider that in this paper.

3.3. Buffer Circuit Design

The area and delay performance of the switching elements is sensitive to the transistor-level circuit design of the multiplexers and buffers. In this section, we describe the methodology used to optimize the switch designs for bidirectional wiring (tristate design) and single-driver wiring (non-tristate design).

In the design process below, the switches are optimized for a wire length of 4 CLBs in TSMC's 0.18μm technology. We assumed that each CLB is 100μm on each side, but our area model suggests that 120-150μm is more typical.

3.3.1. Detailed Design Circuit models used for HSPICE simulations of tristate and single-driver switch elements are shown in Figure 8. The components in each circuit (from left-to-right) are: sense buffer, multiplexer, tapered driver, optional tristate control, wire model, sense buffer loads, optional driver loads, and an optional level-restoring device.

For area reasons, we implement multiplexers as a tree of minimum-size NMOS pass transistors. This allows the use of encoded select lines, reducing SRAM usage. Since reconfiguration is rare and slow, we assume true and complemented outputs are available from a 6-transistor SRAM cell. To reduce unnecessary toggling power, we assume multiplexers are large enough that one input can be connected to ground. When optimizing transistor sizes, tristate switch simulations use a 4:1 multiplexer and single-driver switch simulations use an 8:1 multiplexer. For the architectures considered in this paper, mux sizes up to 16:1 will be employed.

The tapered driver is built using three inverter stages. The PMOS and NMOS devices use different widths (W_p, W_n) to optimize for delay. The first stage must detect a reduced-swing signal (due to the use of pass transistors), so it uses $W_p/W_n = 1/3.5$ to lower the switching threshold. The second and third stages increase in size to drive the wire load. The final drive stage of the tristate switch uses $W_p/W_n = 1/1$. The final drive stage of the single-driver switch uses $W_p/W_n = 1.4/1$.

The wire load is represented by four RC segments. One segment represents one CLB. Tristate switch elements have additional (inactive) driver loads connected to each of these segments. Both switch elements also have one sense buffer load per segment.

The tristate outputs use pass transistors. This degrades the output-high voltage to $V_{dd} - V_t$. A level-restoring keeper is used on the wire to restore a high voltage to V_{dd} and reduce static leakage current. To avoid drive fights that would degrade delay, a weak pullup is used for the keeper. Hence, it only affects signals which will remain high for a relatively long time (*i.e.*, a few ns).

The sense buffer is a single inverter which must drive a load of three multiplexers. The reduced output-high voltage of tristate outputs also influences sense buffer design. For speed, the sense buffer should switch at a lower voltage of approximately $(V_{dd} - V_t)/2$. Hence, the NMOS transistor is wider than its PMOS partner for tristate switches, *i.e.* $W_p/W_n = 1/3.5$. Single-driver switches use a more traditional $W_p/W_n = 2.4/1$.

3.3.2. Simulation Results The results of HSPICE simulations for tristate and single-driver switching elements are shown in Figure 9. The delay is the total end-to-end delay from the input of the switch to the far endpoint of the wire. Since timing results can be asymmetrical, the worst-case delay in response to a rising or falling input is shown.

In the figure, the x-axis corresponds to the switch size, B. Switch size is the ratio of widths of the NMOS transistor, W_n, in the last stage of the tapered driver relative to a minimum-width contactable transistor, W_{minT}.

The delay results indicate that the fastest design is a single-driver switch containing a 4:1 mux. The speed is 330ps per wire at a switch size of 10–14.

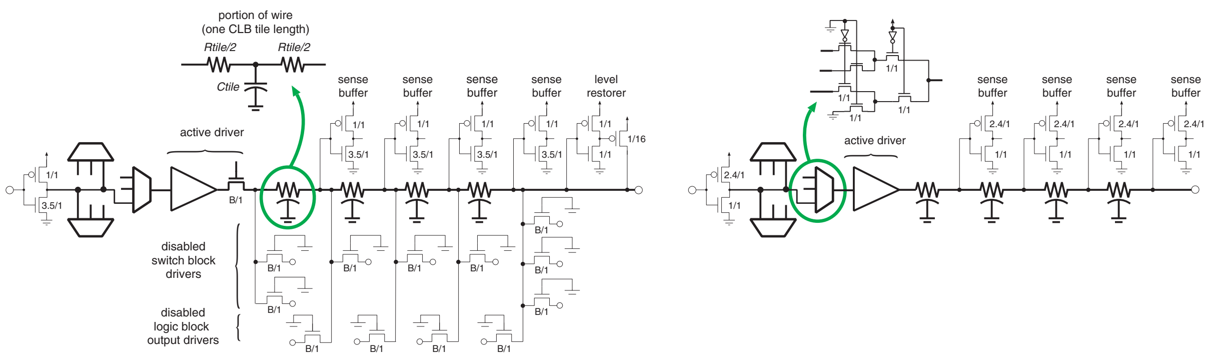

Figure 8. Transistor-level HSPICE models for (a) tristate and (b) single-driver switching elements.

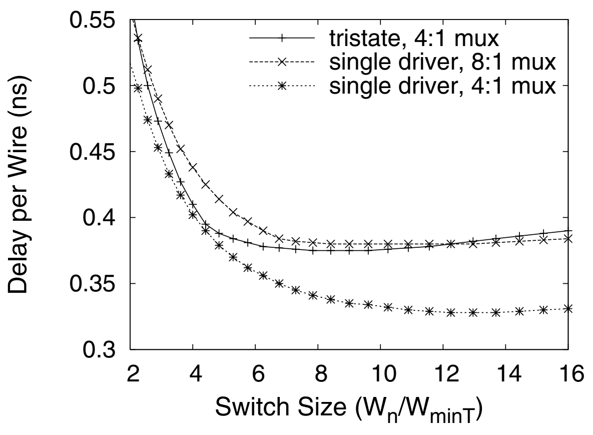

Figure 9. Switch element delay (HSPICE).

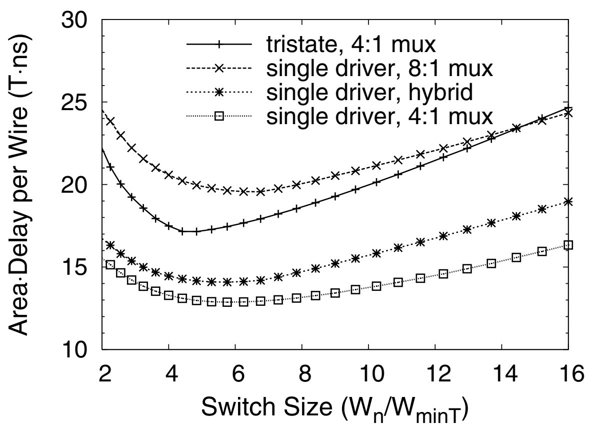

Figure 10. Switch elem. area-delay product.

As expected, the tristate switch is slower at 380ps per wire, but with a smaller switch size of 6–8. The tristate switch performs best with a smaller switch size because numerous inactive drivers also load the wire.

Note that a sufficiently large single-driver switch with an 8:1 mux is almost as fast as the tristate switch with a 4:1 mux.

To determine the best switch size, both delay and area overhead must be considered. Figure 10 shows the area-delay product of these switches. For this graph, delays are taken from Figure 9. Area values are computed using the VPR area model. The area of the switch includes the sense buffer, one multiplexer[1], one tapered driver, the tristate pass transistor, and all configuration SRAM.

The data in Figure 10 suggests the best tristate switch size is 4.4 and single-driver switch size with an 8:1 mux is 6.2. By coincidence, both of these have delays of 390ps. These are the switch sizes used for the remainder of the paper.

It is incorrect to conclude from Figure 10 that tristate switches are better because they have a lower area-delay product than the single-driver switch with an 8:1 mux. The larger multiplexer has more functionality than the tristate switch because it must also absorb CLB outputs. To compare switches with similar functionality, the 'hybrid' curve shows single-driver area-delay using 8:1 mux delay and 4:1 mux area. Another curve shows area-delay using 4:1 mux delay and area. Both of these represent an improvement over tristate switches.

4. Results

The architectural changes and routing switches described earlier have been embedded into VPR by changing its routing graph of the architecture and modifying the appropriate timing and area parameters. The standard VPR place and route algorithms are used to map the 20 largest MCNC benchmark circuits [4] into the FPGA. VPR is used to find a single placement for each circuit. This same placement is used for all subsequent routing experiments.

4.1. Scaling Trends: Expected Performance

The effect of interconnect scaling on FPGA area can be seen in Figure 11. In this figure, the channel width is on the horizontal axis. The right vertical axis is the area of one layout tile containing both the CLB and its adjacent routing. The area units are the number

[1]The other multiplexers are considered to be part of other switches.

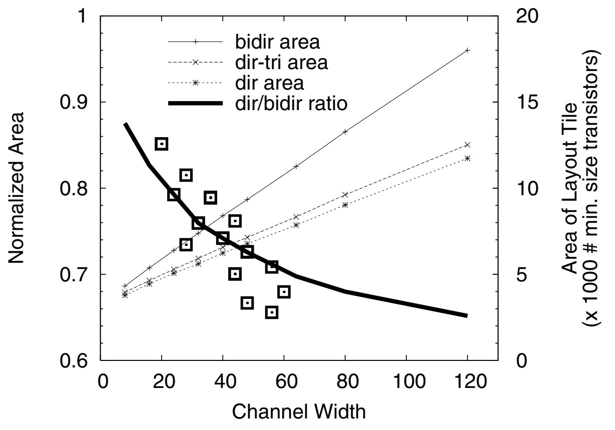

Figure 11. Area savings trend (potential).

of minimum-sized transistors reported by VPR. Area trends are shown for three architectures:

1. The *bidir* architecture is the baseline. Each CLB output is connected to $1/N = 1/6$ of the routing tracks according to Figure 6. The bidirectional, tristate switch element is used.

2. The *dir-tri* architecture uses directional wires. For the same channel width as *bidir*, this eliminates half of the area in the switch block. Each CLB output is connected to $1/6$ of the routing tracks according to Figure 6. The directional, tristate switch element is used.

3. The *dir* architecture employs directional, single-driver wires. All tristate buffers are removed, including those in the switch block and at the CLB outputs. Each CLB output is connected to $2/L = 1/2$ tracks through the switch block muxes shown in Figure 7. The directional, single-driver switch element is used.

From the figure, it is apparent that most of the area savings at a given channel width comes from employing directional wires. However, some additional area savings can be obtained by employing single-driver wires as well.

The *dir* area can be expressed as a fraction of the *bidir* area. This normalized result is shown by the bold curve in Figure 11, and is plotted against the left vertical axis. Assuming that the *dir* and *bidir* routing results of a circuit use the same channel width, this represents the **potential** area savings. The negative slope of this curve indicates that **greater area savings** is obtained **at larger channel widths**.

4.2. Channel Width Results

The lines in Figure 11 indicate **potential** area reduction that is only obtained if the channel widths (after routing) remain the same for all architectures. The boxed datapoints in the same figure indicate the **actual** area obtained after routing the 20 benchmark circuits. There are fewer than 20 datapoints in the figure because some results overlap. Each benchmark was routed in *bidir* and *dir* architectures using the minimum possible channel width.[2] The area obtained with *dir* is normalized to the area obtained with *bidir*. This normalized area is plotted against the channel width obtained with *bidir*. Sometimes *dir* requires more (or fewer) routing tracks than *bidir*. This is why some datapoints lie above (or below) the **potential** normalized curve.

It is notable from Figure 11 that the area savings *increases* as the channel width increases. The amount of savings ranges from 15–34% across the different circuits. Larger circuits benefit more from the use of directional, single-driver wiring.

The relative changes to channel width for each individual circuit can be seen in Figure 12. These results are normalized to the baseline *bidir* architecture results, so a value less than 1.0 indicates a reduction. In the graph, *dir-tri* results are shown with shaded bars and *dir* results are shown with solid bars.

The channel width results show that *dir-tri* requires up to 20% more tracks per channel. With one circuit (spla), it uses 17% fewer tracks. Some of the circuits with the largest increases (bigkey, dsip, s298, tseng) have the smallest channel widths. More importantly, except for one circuit (seq), all of the observed increases can be explained by quantization: *dir-tri* must be a even multiple of L tracks but the *bidir* result was an odd multiple. With seq, one quantum ($2L$) of additional tracks were needed. One larger circuit (spla) needed one quantum fewer tracks.[3] Hence, it is not immediately clear that directional wires need more routing tracks per channel.

The channel width results indicate that *dir* requires fewer tracks than *dir-tri*. In circuits where *bidir* used an odd multiple of L, quantization forced 6 cases to round up and use L more tracks (bigkey, dsip, tseng, misex3, s38417, apex4), and 4 cases required L fewer tracks (s298, apex2, elliptic, pdc). In 2 of the 10 remaining cases (ex5p, spla), *dir* used one quantum ($2L$) fewer tracks.

The numerous **decreases** in channel width are unexpected improvements. The channel width decrease for *dir* can be explained by the increased connectivity at the CLB outputs. With the *dir-tri* and *bidir* architectures, each CLB output connects to only 1/6 of the routing tracks. However, the *dir* architecture connects to 1/2 of the routing tracks. Despite the increased connectivity, there is still a *net area reduction* at the same channel width (Figure 11). The reductions in channel width result in further area savings for the *dir* architecture.

[2]As described earlier, the minimum channel widths are forced to be a multiple of L for *bidir* and $2L$ for *dir-tri* and *dir*.

[3]In this instance, the router got "lucky" and found a solution.

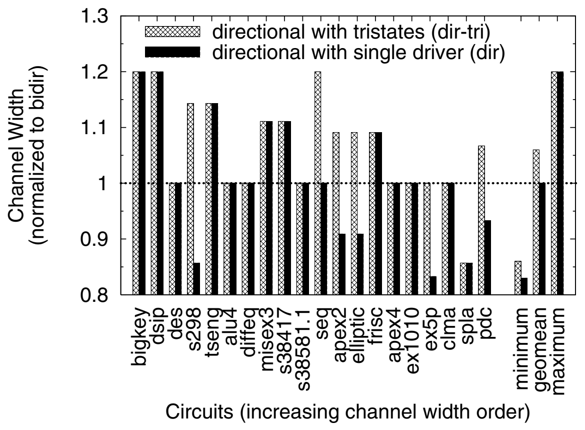

Figure 12. Channel width results.

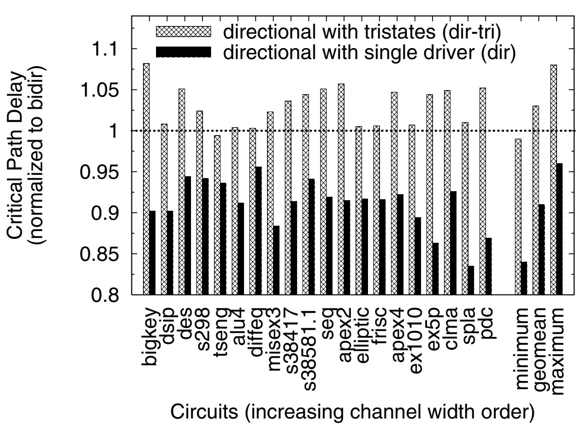

Figure 14. Delay results.

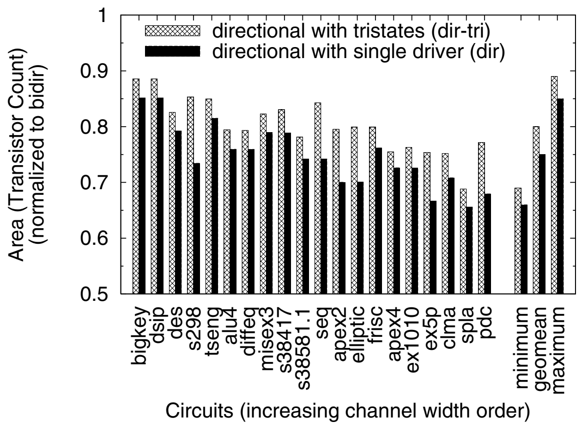

Figure 13. Transistor count results.

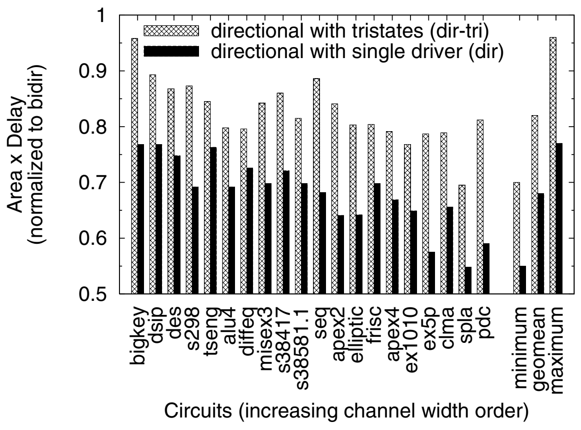

Figure 15. Area-Delay product results.

4.3. Area (Transistor Count) Results

The impact of these different architectures on actual layout area can be estimated by counting the number of minimum-size transistor areas needed to build all of the structures contained within one layout tile. The transistor count results for *dir-tri* and *dir* architectures are normalized to the *bidir* architecture and shown in Figure 13.

These results indicate that, on average, the *dir-tri* architecture produces a 20% area savings (11–31% range). With the *dir* architecture, the savings increases to 25% (15–34% range). Generally, larger savings are obtained from circuits with large channel widths (ex5p, clma, spla, pdc) and from those with channel width reduction (s298, apex2, elliptic). Intuitively, circuits which experience a channel width increase (bigkey, dsip, tseng, misex3, s38417, frisc) also experience the least savings.

The reduction in transistor count leads to a reduction in the length of one CLB layout tile. The average tile length shrink is 14% (8%–19% range), shortening the physical length and capacitance of all interconnect wires by a similar amount. This improves both delay and switching power.

In addition to the tile length shrinkage, the *dir* architecture wires are not loaded with numerous tristate drivers. This results in a further 27% reduction to wire capacitance. When combined, directional, single-driver wires have 37% less capacitance than bidirectional wires. This is an important reduction for switching power and delay.

4.4. Delay Results

To assess the impact of the new routing switches on delay, we re-routed the benchmark circuits with an increased channel width. The new channel width was determined by adding $2L$ tracks to the minimum width required to route with the *dir* architecture. The same increased channel width was used for all three architectures. Critical path delays reported from VPR are normalized to the *bidir* results.

The delay results are shown in Figure 14. The *dir-tri* architecture suffers from a delay increase of 3% on average (-1 to 8% range). The increase may be due to a directional bias with CLB outputs: when designing CLB output connections, VPR picks any $1/6$ of the wires and is not aware of the direction. Due to the tristate CLB output structure, *bidir* and *dir-tri* also include delay reductions of 2–10% from fanout.

The *dir* architecture does not suffer from fanout degradation and outputs connect to equal tracks in both directions. It also has lower wire capacitance due to tile length shrinkage. This results in a 9% average (4–16% range) critical path delay reduction.

4.5. Area-Delay Product Results

The result of multiplying the area and delay figures is shown in the area-delay product results of Figure 15. This figure indicates an average reduction of 32% in area-delay. The reduction ranges from 23% to 45% for individual circuits.

4.6. Discussion and Summary

Bidirectional wires in an FPGA present certain disadvantages: only half of the bidirectional drivers are used in a programmed FPGA, and tristate outputs degrade signal quality. In contrast, directional, single-driver wiring leads to a reduction in area, delay, and capacitance. It also eliminates the problem of degraded signals.

For practical reasons, our results have imposed quantization constraints on the channel width. Specifically, the channel width must be multiple of the wire length $2L$ (L) for bidirectional (directional) wiring.[4] This is necessary so a device can be built from a single layout tile. The $2L$ quantization sometimes causes a small increase in routing channel widths, but there was always a net area reduction.

This paper has presented a new switch block for directional wiring. The primary difference is the removal of drivers connecting wire midpoints in one orientation (horizontal or vertical) to wire midpoints in the other orientation. This was done to make the transition to a single-driver easier. Wire midpoints are important opportunities for a signal to turn, so they are replaced with the ability to connect from the midpoints of one wire to the two wires beginning in the opposite orientation. Even with the directional restriction, benchmark circuits are able to route with a similar number of routing tracks as the bidirectional switch block.

With single-driver wiring, wires must be directional and CLB outputs cannot use tristate connections. Instead, CLB output pins can connect to the beginning of wires in the two switch blocks adjacent to the connection block containing the pin. This imposes a new connectivity restriction on the CLB output pins. Previous work [2] connects each of the N CLB outputs to any of the $1/N$ wires. This work connects each CLB output to the $2/L$ wiring tracks originating in the channel segment. With $L = 4$ and $N = 6$, the *dir* architecture connects to 3 times as many routing tracks. Despite the connectivity increase, there is always a net area savings. Some circuits can even route with fewer routing tracks than the bidirectional architecture, leading to an even better area savings.

[4]It is possible to add a quantum of L directional wires, but they would all have to travel in the same direction!

5. Conclusions

Overall, an average area savings of 25% is realized with directional, single-driver wiring. The savings for individual circuits ranges 15–34%. Delay is improved on average by 9% with a range of 4–16%. Area-delay product savings is 32% on average, and ranges 23–45%. Circuits with wider interconnect channels obtain better savings. Further, wiring capacitance is reduced by 37% due to reduced switch loading and physical wire length shrinkage.

The results in this paper strongly advocate the use of directional, single-driver wiring. There is virtually no channel width impact and there is considerable area savings. Although one could mix bidirectional and directional wires in the same architecture, there seems to be no advantage to doing this.

Further delay improvement may be possible with single-driver wiring through continued circuit design efforts, such as flattening of the pass-transistor tree used in the multiplexer. As well, additional area savings (and possibly delay savings) might be realized by reducing CLB output connectivity or reducing the number of midpoint connections.

6. References

[1] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. In *Int'l Symp. on FPGAs*, pages 3–12, 2000.

[2] V. Betz and J. Rose. Cluster-based logic blocks for FPGAs: Area-efficiency vs. input sharing and size. In *IEEE Custom Integrated Circuits Conference*, pages 551–554, Santa Clara, CA, 1997.

[3] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Boston, 1999.

[4] Collaborative Benchmarking Laboratory. *LGSynth93 suite*. North Carolina State University. `http://www.cbl.ncsu.edu/`.

[5] G. Lemieux and D. Lewis. Using sparse crossbars within LUT clusters. In *Int'l Symp. on FPGAs*, pages 59–68, Monterey, CA, February 2001.

[6] G. Lemieux and D. Lewis. Circuit design of FPGA routing switches. In *Int'l Symp. on FPGAs*, pages 19–28, Monterey, CA, 2002.

[7] G. G. Lemieux and D. Lewis. *Design of Interconnection Networks for Programmable Logic*. Kluwer Academic Publishers, November 2003.

[8] D. Lewis, V. Betz, et al. The Stratix routing and logic architecture. In *Int'l Symp. on FPGAs*, pages 12–20, Monterey, CA, February 2003.

[9] E. S. Ochotta, P. J. Crotty, et al. A novel predictable segmented FPGA routing architecture. In *Int'l Symp. on FPGAs*, pages 3–11, February 1998.

[10] Xilinx, Inc., San Jose, CA. *Online Datasheets*, 2004.