Circuit Design of Routing Switches

Guy Lemieux
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada   M5S 3G4
lemieux@eecg.toronto.edu

David Lewis
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada   M5S 3G4
lewis@eecg.toronto.edu

ABSTRACT

This paper examines circuit design of buffered routing switches in symmetrical, island-style FPGAs. The effects of switch size, tile length, level-restoring, and slow input slew rates are examined. Two new fanin-based switch designs are used to eliminate nearly all of the increase in delay that arises from fanout with a previous switch design. Alternating between buffers and pass transistors is shown to improve connection delay without fanout by 25%. To take advantage of this, we propose schemes to replace some buffers with pass transistors to simultaneously reduce area and delay. Routing a suite of MCNC benchmark circuits shows that 14% in area·delay, or 7% in delay can be saved using the new switch schemes. Alternatively, approximately 13% in area can be saved with no degradation to delay.

1. INTRODUCTION

FPGA interconnect is often based on programmable tri-state routing switches. The switches must be larger than minimum size to overcome the significant wire capacitance and achieve high performance delay results. If these switches are sized too large, they waste area and contribute to excess capacitance as well.

One way of building routing switches is to use a single multistage buffer which drives one or more pass transistors in parallel. To drive a signal from one wire to another, one of the pass transistors is turned 'on' by a controlling SRAM bit. If a signal must fanout to more than one wire, multiple pass transistors are turned 'on'. When this is done, however, each branch will be driven more slowly than if only one wire was being driven. This increase in delay due to fanout can become quite significant; we have observed increases of more than 100% to individual nets.

One way to avoiding this problem is to provide each branch with its own buffer [1]. However, the area with this approach is prohibitive: each of these buffers is about three times larger than a pass transistor, and there are a large number of them in every FPGA.

This paper proposes two new buffered switch designs that avoid this fanout problem and require less area than previous schemes. The new switches are demonstrated to produce both faster and smaller FPGAs in VPR. Additionally, it investigates numerous transistor-level details encountered in routing switch design such as the the effects of slow-rising inputs, and whether multiple switch sizes are necessary during architectural exploration.

To improve area and delay further, different methods of replacing buffered interconnect wires with a mixture of pass transistors and buffers are examined. For example, one method is the use of a new switch block that can always steer signals through a sequence of given switch types, even in the presence of turns.

1.1 Related Work

There is little published work on circuit design of routing switches. Dobbelaere [2] proposed a novel, self-timed circuit that speeds signals using pass transistors, but it has metastability implications. Circuit design issues for building the LEGO FPGA are described in [3]. Work by Khellah [4] touched on pass transistor sizing. Betz [1] has shown that buffers at 5 times minimum size and pass transistors at 10 times minimum size make low area·delay interconnect in $0.35\mu m$ technology. Some of that work is extended here in greater detail using $0.18\mu m$ technology.

This work proposes using fanin-based switches to eliminate large fanout delays. The Alexander architecture [5] used mux-based switches that can directly accept logic outputs. Our new switches appear to be similar, but we quantify the area and delay advantages.

This work also investigates using fine-grained switching between buffers and pass transistors to save area (versus fully buffered) and to achieve low delay for long connections. In comparison, the Xilinx XC4000EX [6] switch block limits pass transistor delay by using an optional buffer, while the Virtex architectures [7] use fully buffered interconnect. In [8], Sheng adds pass transistors between buffered routing tracks and pass transistor tracks, resulting in reductions of 10% to delay and 6% to area·delay. Our work only replaces existing buffers with pass transistors to produce area and area·delay reductions of roughly 13%.

The remainder of this paper is organized as follows. In Section 2, the methodology is described. Section 3 examines routing switch design in transistor-level detail. The new fanin-based switches are described and evaluated in Section 4. Section 5 examines new ways of combining buffers and pass transistors in the routing and Section 6 concludes.

2. METHODOLOGY

This section describes the methodology used for the HSPICE simulations in Section 3 and the CAD flow in the later sections.

2.1 HSPICE Circuit Simulation

All HSPICE results use "typical" process models of TSMC's $0.18\mu m$ technology. Delay measurements are taken when the signal passes through $(V_{dd} - V_t)/2 = 660$mV at both ends. Only the

worst-case of the rising or falling delay times are used, and voltage swings are always begun at full rail voltage. To account for slow input slew-rate effects, step inputs were conditioned by passing the signal through two identical circuits and measuring delay of the second one.

Metal wires are assumed to be implemented in metal-3 using minimum-width, at twice the minimum spacing. A logical wire length of four clustered logic blocks (tiles) was assumed for all wires. Based on a cluster of four 4-LUTs, we estimated a tile length of 116μm and used it throughout the HSPICE modelling.

2.2 VPR Routing Tool

Elmore delay RC parameters were extracted for the routing switches described in this paper, and the different switch organizations and switch block topologies were implemented in a modified version of the VPR 4.30 [9, 10] routing tool. The VPR timing model was modified to degrade the delay of switches under fanout during routing and timing analysis.

Twenty of the largest MCNC benchmark circuits were mapped into 5-input LUTs (k=5), packed into clusters of six LUTs with 17 inputs, and placed once. Then, each circuit was routed in the baseline architecture (described below) to determine the minimum number of tracks required to route it, W_{min}. This was repeated for clusters of 4- and 6-input LUTs, using 14 and 21 inputs to the cluster, respectively.

The performance of each specific architecture is evaluated by *rerouting* each circuit using $W_{min} + 20\%$ routing tracks. This represents the low-stress situation usually encountered in practise, since designers are seldom comfortable operating near the edge of routability. All area and delay results are reported from the completion of this low-stress route. The rerouting is necessary for two reasons. First, some architectures differ slightly in switch block topology. Second, the timing-driven router should reroute to make appropriate delay-oriented decisions when forming connections with the different switch types.

All results are reported as geometric averages for the 20 benchmark circuits. The area of a circuit is computed as the cluster tile size times the number of logic clusters used. Tile size is determined by counting the number of minimum-width transistor areas used by each transistor in the cluster, including the routing. Wire RC values are determined from the tile size (length). Delay results are computed using the critical-path Elmore delay model in VPR.

Due to extensive circuit redesign, such as removing gate boosting, VPR-computed delay times in this paper are slower than earlier publications [8, 10] using the same technology.

2.3 Baseline Routing Architecture

The baseline routing architecture uses only length 4 wires, with half the tracks connected by size 16 pass transistors, and half connected by size 6 buffers.

The new architectures will use new types of buffered switches and replace some buffers with size 6 pass transistors. We should emphasize that these changes only affect the half of the tracks containing buffers; the other half always use size 16 pass transistors.

3. SWITCH DESIGN

This section investigates circuit design issues of routing switches including leakage current, buffer construction, alternating between buffers and pass transistors, and transistor sizing.
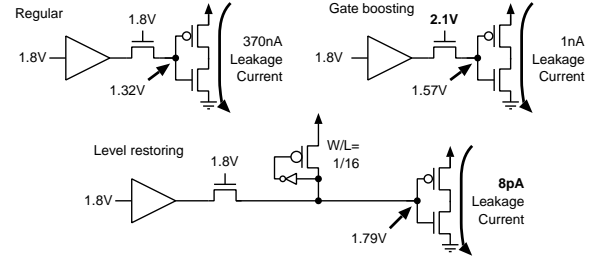

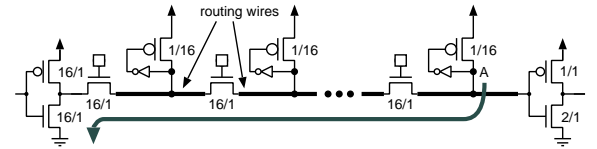**Figure 1: Level-restoring circuit to reduce leakage current.**

Figure 2: The level-restoring pulldown problem.

3.1 Pass Transistors with Level-Restoring

Pass transistors are attractive as routing switches because they require very little area: one pass transistor forms a bidirectional switch and requires only one SRAM control bit. They are very fast for short connections, but delay grows quadratically when they are placed in series. Widening a pass transistor makes it faster by reducing its on-resistance, but making it too wide creates a self-loading problem. Choosing the right transistor width is important for good delay and area-delay performance, and will be discussed later in Section 3.3.

3.1.1 Leakage Current

One drawback of using NMOS pass transistors is that they cause *leakage current* in downstream buffers when passing a logic-high voltage. The steady-state output voltage for such devices is approximately $V_g - V_t$, where V_g is the gate voltage and V_t is the threshold voltage of the device which has been increased in magnitude due to the body effect. This produces a *weak-1* instead of a *strong-1*, causing both transistors of downstream buffers to be partially on. Significant leakage current and static power dissipation results.

The case for 0.18μm is shown in Figure 1. A 1.8V input is reduced to 1.32V after one pass transistor, resulting in 370nA of leakage current in each downstream buffer. When gate boosting is employed [11], an increased gate voltage of 2.1V reduces leakage to 1nA. Gate boosting has been assumed in previous publications [1, 10, 12], but device reliability problems with this technique arise in 0.18μm and below due to the thin gate oxides.

As an alternative to gate boosting, the level-restoring circuit [13, 14] shown in Figure 1 can be used to pull a weak-1 signal into a strong-1. This circuit involves positive feedback of a sense inverter driving the gate of a PMOS pullup (with a long channel length). When a weak-1 is present, the sense inverter begins to turn on the pullup by driving a low signal on its gate. In turn, this increases the voltage of the weak-1 until the pullup has restored the voltage to V_{dd}. One level-restoring circuit is needed on every wire that: *i)* is likely to be driven by a pass transistor, and *ii)* drives a significant amount of regular CMOS logic. All HSPICE simulations in this paper include one level-restoring circuit on every interconnect wire.
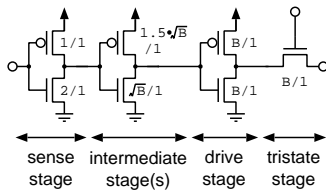
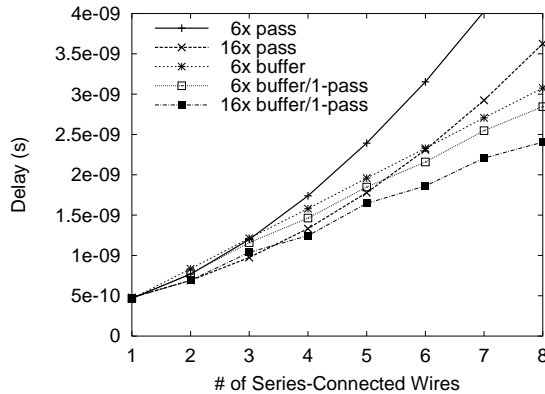**Figure 3: Multistage buffer with (optional) tristate output.**

Figure 4: Connection delay with no fanout.

3.1.2 Level-Restoring Circuit Size

A very strong pullup can quickly restore full voltage to the wire, but this may cause problems. Since the pullup is always on while there is a strong-1 on the wire, it hinders the ability of another driver to pull the signal back to V_{ss}, increasing signal fall time on the wire. FPGA interconnect exacerbates this problem, and may lead to a stuck high state. Figure 2 highlights this *level-restoring pulldown problem*, where the distant node A must be pulled low through a long chain of pass transistors. Since the pass transistors and wires have significant resistance, a voltage divider is formed at node A. If the voltage there cannot be pulled below the switching threshold of the sense inverter, the wire will be stuck high.

The pulldown problem can be addressed by simply preventing connections that would exhibit it. For example, an FPGA router might prohibit the formation of some types of connections, such as very long ones or those with significant fanout. Alternatively, the FPGA architecture might be designed to make such connections impossible to form or easily detected and avoided by the router. For example, carefully placing buffers after every eight pass transistors may provide enough isolation that the worst-case can always be pulled down. For our benchmark circuits, the required FPGA sizes are small enough that using weak pullups appeared to be sufficient. However, a more robust solution is required in the future.

While sizing the level-restoring pullup length, L_p/L_{min}, we examined delay through long chains of N pass transistors of size $W_n/W_{min} = 10$ connecting $N+1$ wires.[1]

Very long chains ($N > 32$) could not be pulled down when the pullup length was sized $L_p/L_{min} < 10$. Chains with $N \geq 16$ wires have their fall times become critical when $L_p/L_{min} \leq N$. This can be quite severe, *e.g.*, at $N = 32$ with $L_p/L_{min} = 10$, fall time is 2.5 times the rise time. We decided to use pullups with $L_p/L_{min} = 16$

---

[1] Throughout this paper, we adopt the notation that L_p is the length of a PMOS device, $L_{min} = 0.18\mu m$, W_n is the width of an NMOS device, and W_{min} is the minimum *contactable* diffusion width.

to compromise between area and pulldown complications. At this length, it takes roughly 50ns to fully restore the routing wire to V_{dd}, and more than 40 series-connected wires can be pulled down.

3.2 Buffer and Buffer/Pass Circuit Design

When long connections are required, pass transistors are unsuitable due to quadratic delay increases. Instead, the linear delay growth of buffered routing switches make them essential for use in large FPGA's. Unfortunately, buffers are slower for short connections and require 2–4 times more area than pass transistors.

The advantages of both switch types can be gained by alternating between a buffer and N pass transistors, a concept we call *buffer/N-pass* switching. Some architectures which support this buffer/pass switching will be examined in the next section, but here we consider the circuit design of the buffers themselves.

Figure 4 gives end-to-end delay when through one to eight routing wires. The alternating buffer/1-pass switches are able to capture the best delay characteristics of both buffers and pass transistors, but this is only for a connection without fanout. For clarity, certain switch sizes were omitted from the figure but their performance is worth mentioning. For example, a size 10 buffer gives the best buffered delay, but this is matched by size 6 buffer/1-pass switch which obviously requires less area. As well, a size 11 buffer/2-pass switch achieves similar delay to the size 16 buffer/1-pass switch.

3.2.1 Buffer Construction

Large buffers are formed by staging multiple inverters as shown in Figure 3. The input drives the first inverter, or sense stage, and the drive stage produces the final output. Intermediate stages, if any, scale up in size by the same fixed factor to reach the drive stage. A tristate buffer is formed by adding an NMOS pass transistor to the drive stage. Other tristate buffer designs with improved drive ability, such as those in [1, 14], were not considered because they require more area.

For example, when constructing a three-stage buffer of size B, the intermediate stage is size $\sqrt{B}$ and the final and tristate stages are both size B. A size B stage uses an NMOS device of width $W_n = B \cdot W_{min}$, and a PMOS device of width $B \cdot W_{min} \cdot W_p/W_n$.

We found $W_p/W_n = 1.5$ to produce minimum delay through multistage buffers, and used this in the intermediate stages. For the sense and drive stages, these ratios were carefully selected in tandem as part of the overall design approach to minimize delay. First, a broad search was done of a number of parameters to explore the design space. Second, the sensitivity of switch size on our tile length assumption was tested. Third, two particular buffer sizes were selected and iteratively optimized to produce a final design. Fourth, the magnitude of delay caused by slow input slew rates was examined to see if routing and timing analysis tools must account for it. Each of these steps will be explained further below.

3.2.2 Broad Search

The first step involves a general sweep, varying drive stage W_p/W_n between 1 and 2, size B from 2 to 64, and the use of 2 to 4 inverter stages driving 1 to 16 series-connected pass transistors and wires. The best delay-per-wire switches were size 14–16, driving two or three wires with $W_p/W_n = 1.0$. The best switch-area·delay per wire switches were similar, but used a smaller size of 6–8. These switch size results were used to guide the the following optimization and verification steps.

3.2.3 Tile Length

The estimated tile length used was $L_{tile} = 116\mu m$, for a total wire length of $4 \cdot 116 = 464\mu m$. Since a variety of larger tile sizes were to
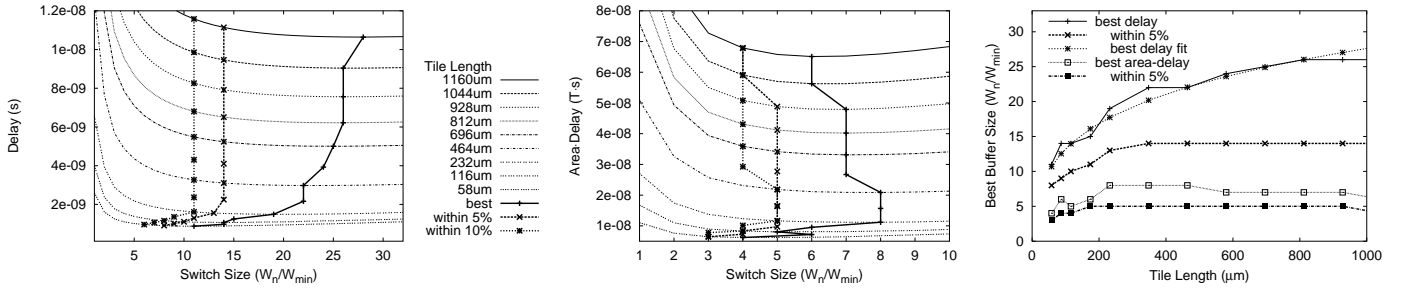
Figure 5: Effect of tile length on performance of a buffer-wire-pass-wire connection.
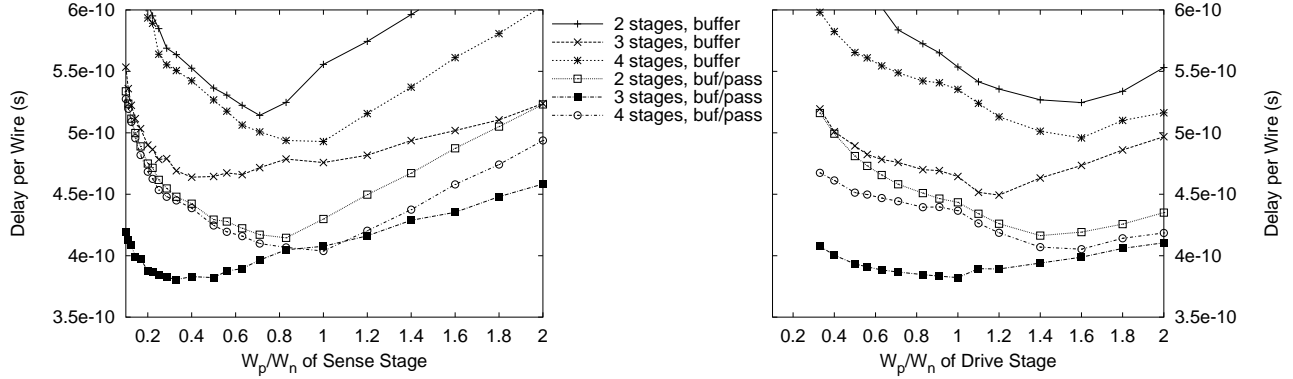

Figure 6: Adjusting a) the sense stage and b) the drive stage to optimize delay-per-wire with a size 6 switch.

be used, it was necessary to verify the sensitivity of the best switch size on L_{tile}.

A buffer/1-pass connection was simulated for a variety of tile lengths. The delay and area·delay product results are shown in Figure 5. In the first two graphs, the switch size B is varied to generate one curve per L_{tile}. The lowest points for each curve are connected together by the bold curve labeled **best**. Two additional bold curves trace the switch sizes that result in being **within 5%** and **within 10%** of the best delay or area·delay. In the third graph, the same best and within 5% data are replotted as a function of tile length.

It is apparent from flatness of each delay curve in Figure 5 that, beyond a certain point, delay is insensitive to the switch size chosen. The best possible delays are achieved by scaling the switch size weakly with tile length, and can be approximated by

$$\text{switch size} = 9.8 \cdot L_{tile}^{0.2} - 11.4$$

for length-4 wires, where L_{tile} is given in μm. However, the figure also shows that a constant switch size of 11 or 14 reaches within 5% or 10% of the best delay, respectively, for tile lengths of 230μm and beyond. Hence, switch scaling is unnecessary for larger tiles.

When the same analysis is conducted on the switch-area·delay product, a fixed switch size of 4–6 is seen as effective for a wider range of tile lengths. However, one difference from before is that the best size actually drops for the longer tile lengths. This is because longer tile lengths, near a switch size of 5, have less delay sensitivity to switch size than the medium tile lengths. This causes the area penalty of the larger switch to be greater than the delay improvement, making the smaller switch size more attractive.

Data in [11] also suggests that the best switch size is insensitive to *logical* wire lengths of 4, 8 and 16 tiles. Although a longer logical length implies more switch loading, wire capacitance dominates. Hence, the effect of tile length and logical length should be similar: they both impact physical wire length and increase its RC.

The ability to use a single switch size for a wide range of architectures greatly simplifies FPGA research. For example, one may construct practical area and delay models based on layout experience without the effort of varying buffer designs for different sizes. It also suggests that previous research which scaled switches linearly with tile size, such as [12, 15], may have slightly over-penalized larger cluster sizes.

3.2.4 Adjusting Sense and Drive Stages

With the understanding that a fixed switch size is sufficient for a broad range of architectures, we iteratively optimized the W_p/W_n of buffer sense and drive stages of size 6 and size 16 switches. This was done for both buffered-only and buffer/1-pass connections.

The optimization began by fixing the drive stage width ratio at $W_p/W_n = 1.5$ and varying the sense stage W_p/W_n from 0.1 to 2. When $W_p/W_n < 1$, the PMOS transistor was fixed at minimum width and the NMOS transistor was widened. Delay-per-wire curves similar to Figure 6a indicated that minimum delay would be reached when the NMOS transistor was minimum size for 4-stage buffers, or slightly wider for 2- and 3-stage buffers (roughly 1.2 and 2 times W_{min}, respectively). In all cases, a minimum size PMOS transistor was used. This reflected the need to sense a lower voltage swing caused by the pass transistor V_t loss.

Next, the sense buffer sizes were fixed at their best values and the drive stage W_p/W_n was varied from 0.3 to 2. Results such as those in Figure 6b indicated that 2- and 4-stage buffers required a stronger PMOS driver with $W_p/W_n = 1.5$, but $W_p/W_n = 1.0$ was sufficient for 3-stage buffers.

After selecting the best drive stage transistor sizes, these values were fixed and another pass was made to re-adjust the sense stage, then the drive stage again. A third iteration verified that the results were stable. The graphs in Figure 6 are the final results of this effort
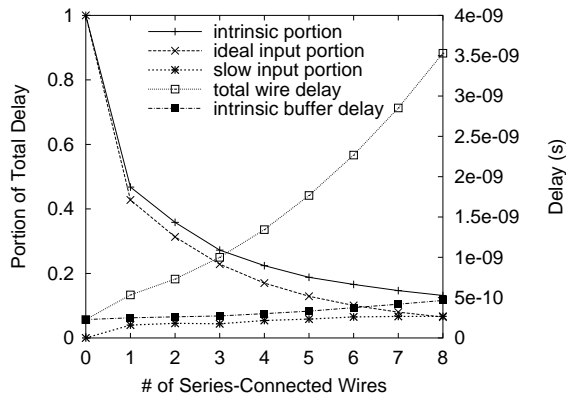
Figure 7: Impact of slow-rising input on delay, size 16 switch.

for size 6 switches. A similar procedure was conducted for size 16 switches, leading to the same transistor width ratios.

Figure 6 indicates that buffer/1-pass switches and 3-stage buffers provide superior delay-per-wire. One reason the 3-stage buffer has lower delay is due to its inverting property. The 2- and 4-stage buffers have worst-case delays caused by slow low-to-high transitions on both the input and output, hence their need for a stronger PMOS pullup to reduce delay. On the other hand, 3-stage buffer delays combine a slow-rising input with fast-falling on the output, for a more balanced effect.

From this data, we decided that all buffer designs should have 3 stages. The sense stage should have a minimum-size PMOS device and a double-width NMOS device, and the drive stage should have equal size NMOS and PMOS devices of size B.

3.2.5 Slow Input Slew-Rate Effect

In VPR, delay through buffers is simplified to a constant value, the intrinsic buffer delay. This is adequate in strictly-buffered interconnect because the input slew rate can be easily determined in advance and included in this delay. However, routing through a series of pass-transistors before being re-buffered will decrease the slew rate, hence increasing the intrinsic buffer delay. We wish to determine whether this *slow slew rate effect* is significant enough to make buffer/pass switches too slow, and whether it is important enough to modify FPGA routers and timing analyzers to explicitly compute it rather than assuming the worst case.

We examined the impact of input slew rate on intrinsic buffer delays by measuring the unloaded buffer delay after an input signal has been degraded by zero through eight pass-transistor connected wires. For the data point at zero wires, the input slew rate uses two minimum size inverters to condition a step input.

Figure 7 presents two sets of results. First, delay results are given for the total delay through up to eight wires and the corresponding intrinsic buffer delay. Although the total delay through the pass transistors is quadratic, the intrinsic buffer delay increases linearly. For the size 16 buffers used in the figure, the delay roughly doubles across the range of inputs. For size 6 buffers, the delay triples across the same range.

Second, the intrinsic delays are plotted as portion of total delay. The **intrinsic portion** curve shows the overall impact of intrinsic delay, including the slew-rate effect. When buffers are placed on every wire, it represents nearly 50% of the delay. As more pass transistors are used, this decreases to about 15% of the total delay. The **ideal input portion** curve removes the slew-rate effect by plotting only the zero-wire input delay as a portion of total delay. The

slow input portion curve shows the difference, *i.e.*, the portion of total delay directly caused by the slew rate effect. Although it increases mildly as the input is degraded, it represents less than 7% of the overall delay. Repeating this with size 6 switches reveals the slew rate effect is highest at three wires or 8% of the total delay, then decreases to 6% at eight wires.

Observations here show that slow input slew rates can increase timing delays by 8%. This is significant enough that timing analysis tools should account for it. However, it is also small enough that first-generation routing tools can probably ignore it.

3.3 Determination of Switch Sizes

Selecting the proper switch size is an important step in creating a low-delay, area-effective interconnect. We simulated the end-to-end delay of a buffer driving one to eight wires connected in series using pass transistors. This represents a wide range from only-buffered wires to primarily pass-transistor connected wires. The delay and area·delay results *per wire* are presented in Figure 8.

In terms of delay, superior results were obtained with buffer/pass switches. Using buffers only, the best sizes of 8–10 resulted in delays of 430ps per wire. This was nearly matched at 450ps per wire by buffer/7-pass using size 16 switches. In contrast, delays as low as 330ps and 320ps per wire were simulated using size 9–16 buffer/1-pass and size 10–16+ buffer/2-pass switches, respectively. These are about 25% faster than purely-buffered interconnect.

In terms of area·delay product, we present two different results. In the second graph of Figure 8, the average switch-area per wire is multiplied by delay per wire. Unfortunately, switch-area per wire unfairly penalizes large switches because it ignores the overhead of the logic cluster. For example, the best buffers are shown to have nearly twice the switch-area·delay per wire of the best buffer/pass switches. In the third graph, an adjustment is made to account for fixed overhead from logic cluster area *and* logic cluster delay. This was accomplished by adding the constants 16T to the switch area and 100ps to delay. These values were chosen to reflect our observations of logic overhead: approximately 45–65% of total area and 20–25% of total delay.

The switch-area results suggest buffer/2-pass and buffer/3-pass switches of size 5–10 is best. In contrast, the adjusted results suggest size 7–12 is best. In both cases, the buffer/1-pass switch is only slightly higher, with the best sizes being 6–10 in the adjusted results. Using the adjusted results significantly alters the choice of best buffer size, from 4–6 to 6–8.

Based on this data, we have chosen to use size 6 and 16 switches in our interconnect. Half of the interconnect tracks will be based on size 6 buffers. In our later experiments, some of these buffers will be replaced with size 6 pass transistors, hence forming buffer/pass switches. The remaining half of the interconnect tracks will be based on size 16 pass transistors only, and will be driven from the logic clusters using size 16 buffers. Using only two switch sizes helps simplify experimental conditions.

The choice of using size 16 pass transistors is probably larger than one would choose from Figure 8 to build an FPGA product. Our choice will result in a very fast and slightly larger interconnect. This was done to make our area and delay improvements *more conservative*. To see this, consider that the pass-transistor based tracks will be held fixed to an area portion that is larger than required. Hence, our area savings from reducing the buffered portion will be understated. Similarly, making the pass transistor portion fast makes it difficult for buffer/pass connections to be faster. This will reduce the need for the delay improvement from buffer/pass switches, again understating any gains we achieve.
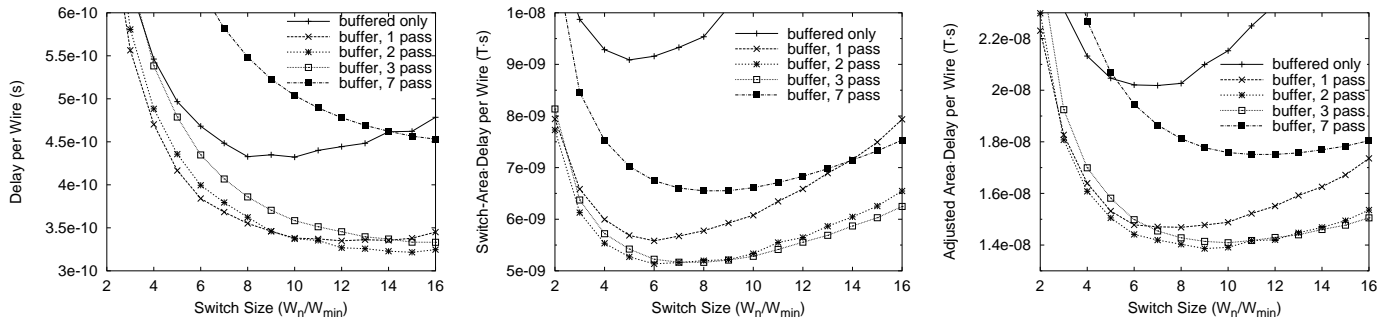
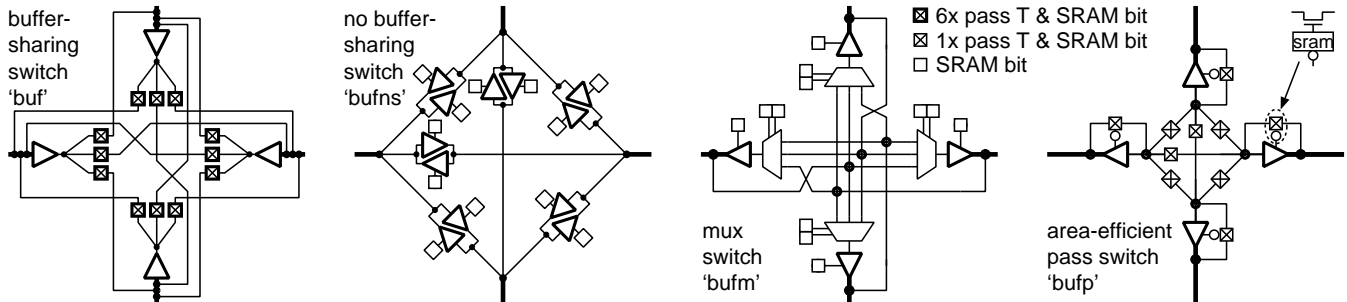**Figure 8: Delay and area-delay per wire for various switch sizes.**

Figure 9: a) Two previous switch types on the left, and b) two new switch types on the right.

4. TWO NEW SWITCH DESIGNS

Historically, VPR has used two different switch types when computing area and delay estimates. Area reports were based on the small, buffer-sharing switch, *buf*, that is shown on the left in Figure 9a. Delay of nets driven by this switch degrade significantly under fanout. In contrast, the switch without buffer-sharing, *bufns*, does not degrade under fanout, but it requires significantly more area. The drawbacks of the two different switch types prompted the creation of two new ones which *i)* do not degrade under fanout, and *ii)* are as area-effective as the *buf* switch.

4.1 Fanin-Based Switches

Two new switch designs, *bufm* and *bufp*, are shown in Figure 9b. The concept behind these new switches is to *pull*, rather than *push*, a signal across the switch block. By changing to a pull, the buffers avoid fanout entirely, and the large pass transistors on the buffer outputs are replaced with smaller ones on the buffer inputs. Hence, fanin-based switches also offer potential area savings.

The differences between *bufm* and *bufp* are as follows. The *bufm* switch assumes a mux-tree structure on the input side, requiring only a few SRAM bits under high fan-in conditions. The *bufp* switch replaces the mux-tree with a flat layer of NMOS pass transistors and one SRAM bit per input. A novel, area-efficient variation of *bufp* is shown in Figure 9. With this arrangement of the input pass transistors, they are used in a bidirectional fashion to significantly reduce SRAM count.

The implementations of *bufm* and *bufp* have similarities. Both switches use similar driving structures, and both use minimum-sized NMOS transistors on the input side. For performance reasons, level-restoring was not done on the internal points of these switches. Level-restoring would require the use of larger NMOS transistors to overpower the restoring pullup. Static leakage currents should be kept low because only a fraction of the buffers on each wire will be unused by the netlist.

To better illustrate the area overhead of each switch type, an area profile is shown in Table 1. The area of each switch type is divided up into number of SRAM bits, large buffers, and large and small pass transistors that are required to connect four wires at a switch block endpoint. Using the transistor area model from [1], this is converted into an area count for two switch sizes.

Although the *bufm* and *bufp* switches have a higher intrinsic delay due to their input structures, this increase is small. The delay per wire of a connection with a fanout of three was measured using HSPICE and normalized to the *bufns* delay in Figure 10. For a wire length of four tiles, *bufp* is 16% slower and *bufm* is only 7% slower. In comparison, there is up to a 110% increase in delay using the *buf* switch under similar fanout conditions.

We should note that early versions of the *bufm* and *bufp* switches used a minimum-sized inverter to isolate the input structures. This increased delay significantly, partly because of the longer path and partly because the switch was no longer inverting. Careful transistor-level design is crucial to performance.

4.2 Output Pin Merging

The input structure of the *bufm* and *bufp* switches is ideal to support larger fanin without a significant increase in area or delay. To take advantage of this efficiency, we connected logic output pins directly into these routing switches. This feature, which we call *output pin merging*, was also used in [5].

Connecting one output this way requires only two additional small pass transistors with *bufm*, or one SRAM bit and one narrow pass transistor with *bufp*. When these switches aren't present, such as in pass transistor switched tracks, the regular method involving a shared buffer, one wide pass transistor, and one SRAM bit must be used. This change does not significantly impact output pin delay, because they require a large buffer structure in both cases.

Switch Type	F_s	Area Profile	Trans. Area (T) size 6	size 16
pass	3	$6S+6P$	57	87
pass	2	$4S+4P$	38	58
bufns	3	$12S+12P+12B$	276	480
bufns	2	$8S+8P+8B$	184	320
buf	3	$12S+12P+4B$	168	276
buf	2	$8S+8P+4B$	130	218
bufm	3	$12S+4P+4B+16p$	156	224
bufm	2	$8S+4P+4B+8p$	124	192
bufp	3	$16S+4P+4B+12p$	176	244
bufp	2	$12S+4P+4B+8p$	148	216
bufp (efficient)	3	$10S+4P+4B+10p$	138	206
bufp (efficient)	2	$8S+4P+4B+8p$	124	192

Key: SRAM (S), large pass transistor (P) and buffer (B), small pass transistor (p). $S = 6T$, $p = 1T$. Size 6: $P = 3.5T$, $B = 13.5T$. Size 16: $P = 8.5T$, $B = 25.5T$.

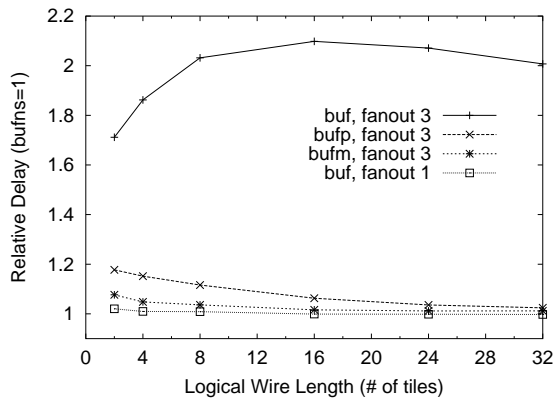**Table 1: Transistor area to connect four wire endpoints.**

Figure 10: Delay per wire under fanout, normalized to *bufns*, size 6 switches.

4.3 Experimental Results

Constant-delay timing models of the *bufm* and *bufp* switches where constructed for VPR based on the worst-case fanin conditions. To model buffer fanout delay, we modified VPR to add an RC node at the drive stage output (before the tristate stage). This change caused delay increases of up 200% or more for individual nets.[2] The changes in net delay increased critical-path delay by 5% on average.

The average critical-path routing delay results for 20 MCNC circuits are presented in Table 2. The *ignoring fanout* columns contain the delays reported by VPR using the original *unmodified* timing model. The *including fanout* columns give the new delay that is computed with the buffer fanout modifications. This is also reported as a percentage in the *increase* columns. For all three LUT sizes, fanout at *buf* switches increased the average critical path by 5%. For individual circuits, this increase was as high as 16%. In contrast, the *bufp* and *bufm* switches limit this increase to 1% on average, or up to 5% for individual circuits. This increase would be zero, except that output pins still use the *buf* switch to connect with pass transistor tracks.

---

[2]To save area, output pins still use a shared-buffer switch to connect to the numerous pass transistor tracks. This higher fanout causes higher delay increases.

There is some variation in the *ignoring fanout* columns for the different switch types. This variation comes from two sources. First, the switches have different intrinsic delays, so we would expect *bufp* to be slower, for example. Second, benchmarks were rerouted for each switch type because the timing-driven router must be able to to make different delay trade-offs such as using pass transistor tracks and changing fanout patterns.

The normalized area results in Table 3 show that the new switches save 2–3% in transistor area. It is interesting that both *bufp* and *bufm* have similar transistor area costs, despite the area profile showing *bufp* to be lower. The main reason for this is that *bufm* saves more area from output pin merging. In terms of delay, *bufm* saves 7% compared to only 2–5% with *bufp*. When combined with the area improvements, area·delay product is reduced by 5% and 9% for the *bufp* and *bufm* switches, respectively.

Although equivalent in area, the *bufm* switch is superior to *bufp* in terms of delay. For this reason, we will focus on using *bufm* in the remainder of this paper.

5. BUFFER/PASS ARCHITECTURES

In this section, numerous routing architectures that allow a signal to be switched by a combination of buffers and pass transistors are presented. We first introduce two switch schemes that alternate between buffers and pass transistors. This concept is generalized to cycle among a collection of N_g different switch types. Then, we describe some less-structured buffer/pass architectures. All architectures considered can be derived from the *baseline architecture* by replacing some of the buffers with pass transistors to save area and to potentially reduce delay.

5.1 Alternating Buffer/Pass Switches

We have identified two ways of replacing buffers with pass transistors so that long connections alternate between the two switch types. In the first scheme, *alt1*, two buffers which are normally "in parallel" but drive opposite directions have one of the buffers replaced. In the second scheme, *alt2*, an entire group of buffers is replaced at every other switch block.

With a small modification to the switch block topology, these changes can be implemented in a single layout tile. Starting with the 4-wire cliques in Figure 9, a twisting of straight connections in track pairs, as shown in Figure 11, is sufficient to implement *alt2*. The *alt1* scheme requires this twist plus different turning connections. Greater circuit-level detail of these schemes, using the *bufp* switch, is shown in Figure 12.

The *alt1* scheme promises greater speed at the expense of slightly higher area. With this arrangement, some wires will be driven by only buffers, while others will be driven by only pass transistors; any long connection will always strictly alternate between being on a buffered wire and an unbuffered one. It is potentially faster because loading occurs close to the buffer source: *i)* all pass-transistor fanout occurs as a signal leaves the buffered wire, and *ii)* all of the diffusion capacitance from unused buffer switches is connected to the same buffered wire. The pass transistor is only used in a direction opposite to the buffer (this was enforced by the router).

In comparison, the *alt2* scheme uses less area because it takes advantage of both switching directions of a pass transistor. This reduces the number of pass transistors and SRAM bits required.

5.1.1 Generalizing: Any Switch Sequence

The switch block changes for *alt1* can be generalized so that long connections will cycle among any sequence of switches. For example, one possible sequence would be a buffer followed by two pass transistors. This strict cycling continues in the presence of turns, provided that the turns are made at wire endpoints only.

LUT	*buf* routing switch			*bufp* routing switch			*bufm* routing switch		
size	ignoring	including		ignoring	including		ignoring	including	
k=	fanout (ns)	fanout (ns)	increase	fanout (ns)	fanout (ns)	increase	fanout (ns)	fanout (ns)	increase
4	16.9	17.8	5.2%	17.3	17.4	0.8%	16.5	16.6	0.6%
5	16.2	17.1	5.3%	16.5	16.7	0.8%	15.8	16.0	0.8%
6	15.4	16.2	5.3%	15.3	15.4	0.6%	14.9	15.0	0.8%

Table 2: Delay increase due to buffer fanout (geometric average of the critical-path (Elmore) delay for 20 MCNC circuits).

Buffer	Trans. Area ($\times 10^6 T$)			Delay (ns)			Area·Delay (T·s)		
Type	k=4	5	6	k=4	5	6	k=4	5	6
unnormalized	3.25	3.34	3.28	17.8	17.1	16.2	0.058	0.057	0.053
buf	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
bufp	0.97	0.98	0.98	0.98	0.97	0.95	0.95	0.95	0.94
bufm	0.97	0.98	0.98	0.93	0.93	0.93	0.91	0.91	0.91

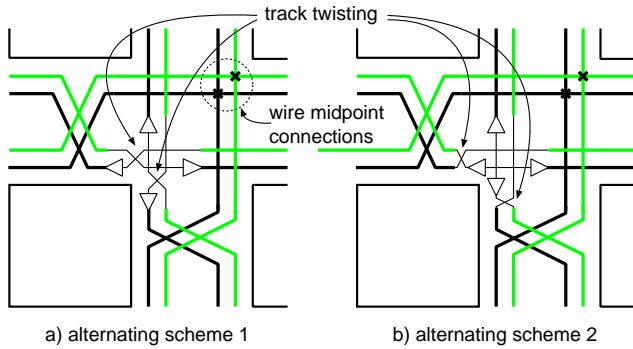**Table 3: Transistor area, delay, and area·delay results using different buffer types.**

a) alternating scheme 1   b) alternating scheme 2

Figure 11: Buffer locations and straight-through connections for two proposed buffer/1-pass schemes.

a) alternating scheme 1
(two tracks shown)

b) alternating scheme 2
(one track shown)

Figure 12: Details of two buffer/1-pass switches using *bufp*.

To accomplish this cycling among a group of N_g switch types, the tracks in a channel are divided into N_g groups and then then interconnected as follows. Tracks in group g are connected to tracks of another group across the S block, $f(g)$. We have adopted the convention that switch type g is used for all connections *to* group g. Different $f(g)$ mapping functions are selected depending on the turn direction, as shown in Figure 13. The figure also provides examples for connecting two or three different switch types. The specific decision of which tracks on the two sides are connected does not matter, provided the tracks are each from the proper group.

5.2 Architectures Considered

In addition to the above two alternating schemes, another approach for replacing buffers with pass transistors is to treat wire endpoint connections separately from wire midpoints. A wire segment spanning L tiles passes through $L - 1$ switch blocks where midpoint connections can be made. Midpoint connections for length 2 wires are shown in Figure 11.

The architectures we explored involve assigning all combinations of different switch types to endpoint and midpoint connections. The midpoint switch types considered were: buffer, pass transistor, *alt1*, or *alt2*. Separately, the endpoint switches types considered were: buffer, pass transistor, *alt1*, *alt2*, strbuf_turnpass, or strpass_turnbuf. The last two schemes involve arranging buffers such that only straight or only turning connections are buffered, and the other connections use pass transistors.
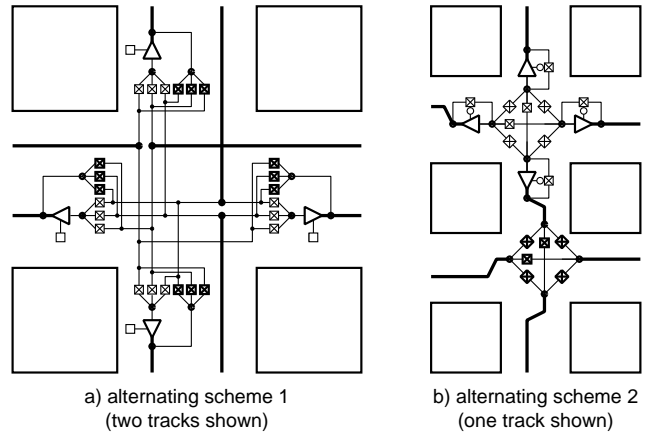
Many combinations of the above switch schemes do not force alternation for long, straight connections. However, delay can still be reduced because most of them do create the opportunity for alternation as a connection executes turns.

5.3 Experimental Results

Experimentation was performed to determine the best way to replace some buffers with pass transistors. The architectures described in the previous section were used in routing experiments to determine the best organization for area, delay and area·delay.

Table 4 presents the main results of the different buffer/pass architectures. The entries in this table are normalized to an architecture using *bufm* switches at midpoint and endpoint locations.

The first set of rows compare the performance of the *buf* and *bufm* switches. In both cases, more area is saved as more midpoint switches are replaced with pass transistors, from *alt1* to *alt2* to *pass*. There is greater savings with 4-input LUTs, since a greater proportion of its area is consumed by the routing. Most of these buffer/pass architectures have higher delay: the *buf* switch increases delay by 8–10%, while *bufm* often increases it by 2–3%. In terms of area·delay, the delay increase of the *buf* switch dominates to result in less efficient architectures (> 1), while the *bufm* area savings produces more efficient architectures (≤ 1).

$$
\begin{aligned}
f_1(g) &= (g+1) \mod N_g \\
f_2(g) &= (N_g - g + 1) \mod N_g \\
f_3(g) &= (g+1) \mod N_g \\
f_4(g) &= (N_g - g - 1) \mod N_g \\
f_5(g) &= (g+1) \mod N_g \\
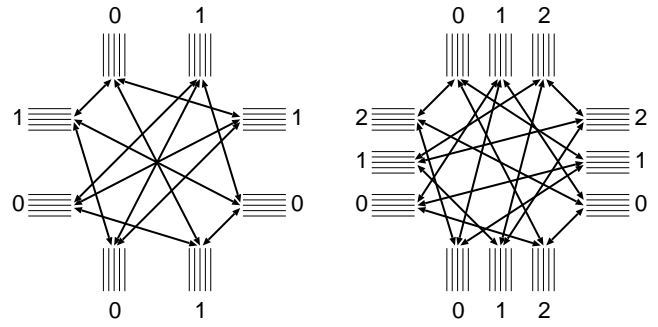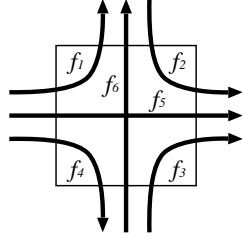f_6(g) &= (g+1) \mod N_g
\end{aligned}
$$

Figure 13: Switch block to evenly cycle through a sequence of switches. The examples on the right cycle among 2 or 3 switches.

The second set of rows evaluate the remaining buffer/pass architectures. Due to its superior delay performance, only *bufm* switch results are included in this portion of the table. As well, endpoint switches using only pass transistors were excluded due to very poor delay results (20–70% increases). In the following discussion, the terminology *alt1-pass* architecture will refer to interconnect using *alt1* endpoint switches and *pass* midpoint switches.

All of the remaining buffer/pass architectures are slower than the *buffered-buffered* architecture. The *alt1-buffered* and *alt2-buffered* versions had the lowest delay increase at 2–3%. Despite being designed to be faster, the *alt1-alt1* organization was only about 1% better than *alt2-alt2*, and *alt1-pass* was up to 3% slower than *alt2-pass*. The *strbuf_turnpass* and *strbuf_turnpass* architectures generated delay results that were typically 10–19% slower.

It was disappointing that the buffer/pass architectures were unable to realize any delay improvement. We constructed simple test circuits with long connections, and these achieved 15–20% delay reduction using the Elmore delay models. We routed the benchmarks with significantly more tracks than required, but this failed to produce a delay improvement. We found that the critical path in each circuit usually included high fanout nodes. Under fanout, the buffer/pass connections would slow down and begin to erase the expected gains. Further investigation is required to improve delay.

Area trends were similar to before. In general, the architectures using *pass* used the least area, followed by those using *alt2*, *alt1*, then *buffered*. Interconnect using *alt2-alt2* was 2% better than using *alt1-alt1*. The *strbuf_turnpass* and *strbuf_turnpass* architectures were unable to save more area than *alt1-pass* or *alt2-pass*, which used the least.

Only a few of the remaining architectures were able to improve area·delay, and only for 4-input LUTs. The best were *alt1-alt1*, *alt2-pass*, *alt2-alt2*. All of the *alt* architectures kept area·delay increases within 6%, compared to a 4–12% increase for the others.

The best architectures are summarized in Table 5. Unlike the previous table, these results have been normalized to the *buf* switch results to illustrate the total area and delay savings realized. The best delay architecture still uses only buffered interconnect, but saves 7% using the *bufm* switch. The best area·delay architecture changes midpoint switches to pass transistors, to achieve a 11-14% savings. The best area architecture saves 8–13% in area yet sacrifices only 1–2% in delay.

6. CONCLUSIONS

Replacing gate-boosting with a level-restoring circuit was an effective way to solve the static power dissipation problem. Attention must be paid to the level-restoring pulldown problem to avoid dangerous stuck-high states. Level-restoring circuits also impact signal falltime, sometimes making it the dominant delay.

Routing buffers were fastest when they were inverting, consisting of 3 inverter stages. A larger NMOS transistor on the input stage helps speed signalling, and a weak PMOS transistor on the drive stage is sufficient to drive through NMOS pass transistors.

We found that slow input slew rates increases intrinsic buffer delays by two to three times. This can account for up to 8% of net delay when buffers are mixed with pass transistors. Timing analyzers and second-generation routing tools should account for these effects.

Testing a wide range of tile lengths for optimum buffer size in terms of area and area·delay produced the unexpected result that a fixed size is probably sufficient to achieve within 5% of the best possible. However, switch sizes should probably be scaled for best delay when small tile lengths of less than 150–200μm are used.

Modelling buffer fanout delay in VPR increased delay results by 5%. Two new switches, *bufp* and *bufm*, were able to practically eliminate all of this increase because they have no fanout. The *bufm* switch was found to be faster than *bufp*, so it was the preferred choice. The new switches are more area efficient than the previous ones, particularly for large buffer sizes and when higher flexibility is required.

Replacing some buffers with pass transistors was performed to create an interconnect capable of alternating between buffers and pass transistors. In doing this, area savings was guaranteed. However, even though alternating between these switches was observed to be 25% faster in HSPICE, no delay improvement was seen in the final routed circuits. A delay increase of up to 5% was typical, but up to 70% was observed for organizations with pass transistors at wire endpoints. Presumably, this increase was caused by fanout loading; the delay improvement was only expected for single fanout nets.

Overall, the architectures of choice are summarized in Table 5. These results have been normalized to the baseline architecture results to illustrate the total area and delay savings realized with the new switch. The best delay architecture still uses only buffered interconnect, but saves 7% using the *bufm* switch. The best area·delay architecture changes midpoint switches to pass transistors, to achieve a 11-14% savings. The best area architecture saves 8–13% in area yet sacrifices only 1–2% in delay. Due to the use of fast, wide pass transistors in the unmodified portion of the interconnect, these results are deemed to be conservative estimates.

7. ACKNOWLEDGEMENTS

Endpoint Switches	Midpoint Switches	Buffer Type	Trans. Area ($\times 10^6 T$)			Delay (ns)			Area·Delay (T·s)		
			k=4	5	6	k=4	5	6	k=4	5	6
unnormalized			3.16	3.27	3.23	16.6	16.0	15.0	0.0523	0.0522	0.0485
buffered	buffered	bufm	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
buffered	alt1	buf	0.99	0.99	1.00	1.08	1.10	1.10	1.07	1.09	1.10
buffered	alt2	buf	0.98	0.99	0.99	1.09	1.10	1.10	1.07	1.09	1.09
buffered	pass	buf	0.94	0.95	0.96	1.09	1.10	1.11	1.02	1.04	1.07
buffered	alt1	bufm	0.98	0.98	0.99	1.03	1.00	1.00	1.01	0.98	0.99
buffered	alt2	bufm	0.97	0.97	0.98	1.02	1.03	1.00	0.99	1.00	0.98
buffered	pass	bufm	0.93	0.94	0.96	1.02	1.03	1.02	0.95	0.97	0.98
alt1	buffered	bufm	0.99	0.99	1.00	1.03	1.03	1.03	1.02	1.02	1.03
alt1	alt1	bufm	0.96	0.97	0.98	1.04	1.06	1.05	0.99	1.02	1.03
alt1	alt2	bufm	0.96	0.96	0.97	1.05	1.06	1.07	1.01	1.02	1.04
alt1	pass	bufm	0.91	0.93	0.95	1.12	1.12	1.12	1.03	1.04	1.06
alt2	buffered	bufm	0.98	0.98	0.99	1.03	1.02	1.02	1.01	1.00	1.00
alt2	alt1	bufm	0.95	0.96	0.97	1.06	1.07	1.07	1.00	1.02	1.03
alt2	alt2	bufm	0.94	0.95	0.96	1.05	1.06	1.06	0.99	1.01	1.02
alt2	pass	bufm	0.90	0.92	0.94	1.09	1.09	1.11	0.97	1.00	1.04
strbuf_turnpass	buffered	bufm	0.99	0.99	0.99	1.06	1.07	1.09	1.05	1.06	1.08
strbuf_turnpass	alt1	bufm	0.96	0.96	0.97	1.11	1.13	1.16	1.06	1.09	1.12
strbuf_turnpass	alt2	bufm	0.95	0.96	0.97	1.12	1.13	1.12	1.06	1.08	1.08
strbuf_turnpass	pass	bufm	0.91	0.92	0.94	1.18	1.18	1.19	1.07	1.09	1.12
strpass_turnbuf	buffered	bufm	1.00	1.00	1.00	1.05	1.05	1.05	1.04	1.05	1.04
strpass_turnbuf	alt1	bufm	0.97	0.98	0.98	1.10	1.10	1.12	1.07	1.08	1.10
strpass_turnbuf	alt2	bufm	0.96	0.97	0.98	1.09	1.10	1.09	1.05	1.07	1.07
strpass_turnbuf	pass	bufm	0.92	0.94	0.95	1.14	1.13	1.16	1.06	1.06	1.11

Table 4: Area, delay, and area·delay results using different buffer types and mixing of pass transistors with buffers.

Criterion	Endpoint Switches	Midpoint Switches	Buffer Type	Trans. Area ($\times 10^6 T$)			Delay (ns)			Area·Delay (T·s)		
				k=4	5	6	k=4	5	6	k=4	5	6
unnormalized				3.25	3.34	3.28	17.8	17.1	16.2	0.058	0.057	0.053
baseline	buffered	buffered	buf	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
best delay	buffered	buffered	bufm	0.97	0.98	0.98	0.93	0.93	0.93	0.91	0.91	0.91
best area·delay	buffered	pass	bufm	0.91	0.93	0.94	0.95	0.96	0.94	0.86	0.89	0.89
best area	alt2	pass	bufm	0.87	0.90	0.92	1.01	1.02	1.02	0.88	0.91	0.95

Table 5: Best architectures compared to the baseline.

8. REFERENCES

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

[2] I. Dobbelaere, M. Horowitz, and A. El Gamal, "Regenerative feedback repeaters for programmable interconnections," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, 1995.

[3] P. Chow, S. Ong Seo, J. Rose, K. Chung, G. Paez-Monzon, and I. Rahardja, "The design of an SRAM-based field-programmable gate array — part II: Circuit design and layout," *IEEE Transactions on VLSI Systems*, vol. 7, pp. 321–330, September 1999.

[4] M. Khellah, S. Brown, and Z. Vranesic, "Modelling routing delays in SRAM-based FPGAs," in *Canadian Conference on VLSI*, pp. 6B.13–18, November 1993.

[5] E. S. Ochotta, P. J. Crotty, C. R. Erickson, C.-T. Huang, R. Jayaraman, R. C. Li, J. D. Linoff, L. Ngo, H. V. Nguyen, K. M. Pierce, D. P. Wieland, J. Zhuang, and S. S. Nance, "A novel predictable segmented FPGA routing architecture," in *ACM/SIGDA Int. Symp. on FPGAs*, (Monterey, CA), pp. 3–11, February 1998.

[6] S. Trimberger, K. Duong, and B. Conn, "Architecture issues and solutions for a high-capacity FPGA," in *ACM/SIGDA Int. Symp. on FPGAs*, (Monterey, CA), pp. 3–9, February 1997.

[7] Xilinx, San Jose, CA, *Online Data Books*, 2001.

[8] M. Sheng and J. Rose, "Mixing buffers and pass transistors in FPGA routing architectures," in *ACM/SIGDA Int. Symp. on FPGAs*, pp. 75–84, February 2001.

[9] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Field-Programmable Logic*, pp. 213–222, 1997.

[10] G. Lemieux and D. Lewis, "Using sparse crossbars within LUT clusters," in *ACM/SIGDA Int. Symp. on FPGAs*, (Monterey, CA), pp. 59–68, February 2001.

[11] V. Betz and J. Rose, "Circuit design, transistor sizing, and wire layout of FPGA interconnect," in *Custom Integraged Circuits Conference*, pp. 171–174, May 1999.

[12] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," in *ACM/SIGDA Int. Symp. on FPGAs*, pp. 3–12, 2000.

[13] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ: Prentice Hall, 1996.

[14] K. Martin, *Digital Integrated Circuit Design*. New York, NY: Oxford University Press, 2000.

[15] A. Marquardt, V. Betz, , and J. Rose, "Speed and area tradeoffs in cluster-based FPGA architectures," *IEEE Transactions on VLSI Systems*, pp. 84–93, February 2000.