# Interconnect Driver Design for Long Wires in Field-Programmable Gate Arrays

Edmund Lee, Guy Lemieux, Shahriar Mirabbasi
*University of British Columbia, Vancouver, Canada*
{ eddyl | lemieux | shahriar } @ ece.ubc.ca

***Abstract* – Each new semiconductor technology node brings smaller transistors and wires. Although this makes transistors faster, wires get slower. As FPGA device designers strive to obtain the lowest possible circuit delays from a given technology node, they must take an increasingly interconnect-focused viewpoint in the design process. In particular, for long interconnect wires, signals now require rebuffering somewhere in the middle of the wire. This paper presents a framework for designing and evaluating long, buffered interconnect wires in FPGAs with near-optimal delay performance. Given a target physical wire length, width and spacing, the method determines the number, size, and position of buffers required to obtain the fastest signal velocity for programmable interconnect. A metric introduced during the design is the "path delay profile", or the arrival time of a signal at different points of a long wire. This method is used to design buffering strategies for interconnect based on 0.5mm, 2mm, and 3mm wire lengths in 180nm technology. These interconnect designs are coded into VPR along with an improved timing analyzer which accurately determines the "path delay profile" arrival times. Using VPR, average critical-path delay is reduced by 19% for 0.5mm wires and by up to 46% for 3mm wires over previous designs given in [1].**

## I. INTRODUCTION

In early FPGA architectures, an interconnect wire was "shared" and could be driven by many possible sources distributed along the length of the wire. Although this made wires general-purpose and "bidirectional", it required several large, tristate buffers per wire, only one of which could be turned "on" for a given programming configuration. As a result, numerous buffers were left unused, which added needlessly to area, capacitance, power and delay.

In comparison, modern FPGAs typically have a *single driver* at the *starting point* of each wire. Instead of tristates, each driver input has a multiplexer to select from many possible sources, creating the flexibility of a switching network. This new organization, dubbed single-driver interconnect [1], results in unidirectional wires. One of the biggest benefits of unidirectional wires is the elimination of bidirectional buffering and tristates.

FPGAs are also among the earliest adopters of deep-submicron semiconductor technologies. Each new technology node brings smaller transistors and wires. Although this makes transistors faster, wires get slower. This makes it increasingly important for FPGA architects and device designers to take an interconnect-focused viewpoint on design. In particular, long interconnect wires behave like an RC transmission line where delay grows quadratically with length. To improve delay, long wires need rebuffering after a certain distance. Each buffer also imposes its own additional delay, so the number of buffers and their positions must be chosen carefully.

For ASIC and custom designs, this problem of rebuffering a signal is often called repeater insertion. For very critical nets, particularly clock trees, this is often done in conjunction with wire sizing. In this environment, the problem of repeater insertion and wire sizing is very complex because there are numerous signals which must be considered, each with different constraints such as large fan-outs to several locations across the chip. To reduce complexity, ASIC tools often rely on fast but low-accuracy Elmore delay estimates.

The problem of repeater insertion for long FPGA interconnect wires has not been widely published [2]. In comparison to the ASIC problem, the FPGA device environment is highly structured. This imposes several simplifying design constraints upon the problem, making it possible to model and solve much more accurately. However, the nature of the problem is also different. In an FPGA device, the location of critical nets (source and sink locations) are not known *a priori* by the designer. An ASIC solution can optimize delay on paths to known critical sinks, and optimize area on the other paths. However, an FPGA solution requires delay to be *optimized to all possible sinks* located along the wire or across the device, yet area cannot be completely forsaken either.

This paper presents a fast, accurate, HSPICE-based framework for designing long, buffered interconnect wires in FPGAs with near-optimal delay performance. Given a target physical wire length, width, and spacing, the method determines the *number*, *size*, and *position* of drivers required to obtain the fastest end-to-end signal speed. A metric called the "path delay profile", shown in Figure 1, is also used. In the figure, we see that a distributed design is faster than lumping buffers at one end because the signal arrives earlier at wire locations before the halfway point. This new method is used to design interconnect for 0.5mm, 2.0mm, and 3.0mm wire lengths in the 180nm technology node.

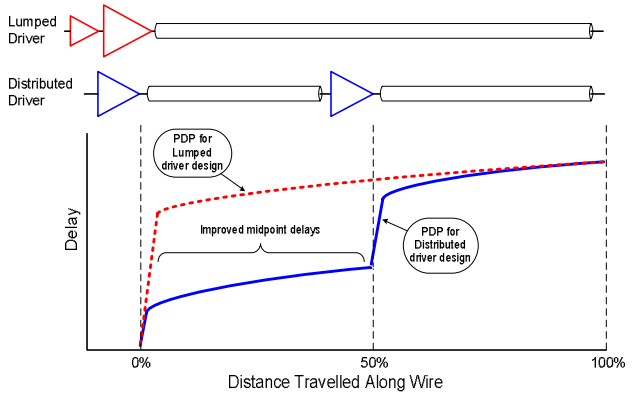



**Figure 1. These path delay profiles suggest distributed buffering is faster because several interior points along the wire receive the signal earlier.**

* This research was partially supported by Micronet R&D and NSERC Discovery Grants. This research has also been enabled by the use of WestGrid computing resources, which are funded in part by the Canada Foundation for Innovation, Alberta Innovation and Science, BC Advanced Education, and the participating research institutions. WestGrid equipment is provided by IBM, Hewlett Packard and SGI.

These interconnect designs are coded into VPR along with a new, more accurate timing analyzer which can determine the "path delay profile" arrival times. The routing algorithm is also modified to make routing decisions based upon improved timing when an *early turn* is made before the end of a wire. Using VPR in this mode, we find that early turns become more common, and that *normal turns* made at the far end of a wire become far less common.

The remainder of this paper is organized as follows. Section 2 provides additional background material. Section 3 describes the interconnect design framework and gives several interconnect design solutions. Section 4 describes the improvements made to VPR in delay modeling and CAD and presents the place-and-route CAD results. Finally, Section 5 presents conclusions.

## II. Background and Problem Formulation

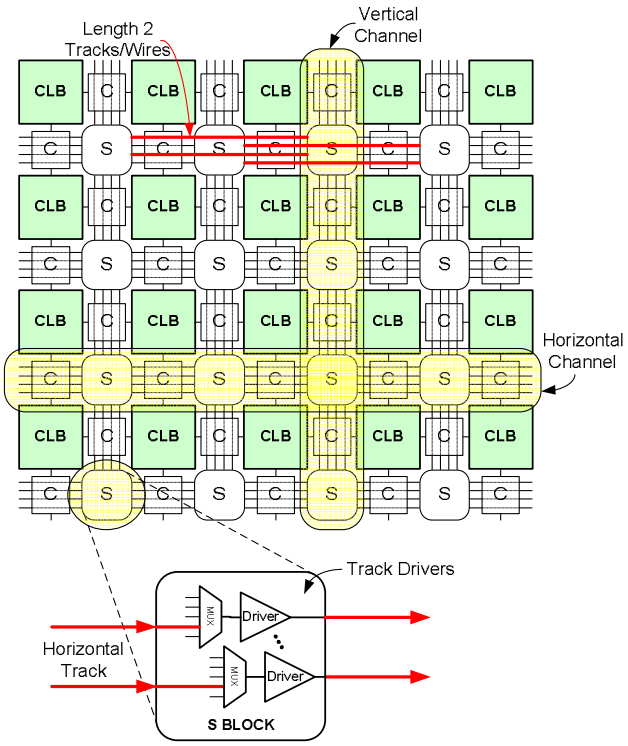



**Figure 2. FPGA architecture and switch block detail.**

The FPGA architecture considered in this paper is shown in Figure 2. Configurable logic blocks (CLBs) containing basic logic elements (BLEs) are surrounded by routing resources interconnected by switch blocks (S) and connection blocks (C). Several *logical* length 2 (L2) wires are shown in the figure; these span 2 CLBs and terminate at S blocks. Total *physical* wire length also depends upon the physical size of the CLB layout tile.

Figure 3 provides additional details of a length 2 wire with surrounding circuitry. Unlike the general ASIC repeater insertion problem, the FPGA interconnect design utilizes a long, straight wire with taps that fan-out along the interior points of the wire. These taps or turns, dubbed "early turns," are always present but may not always be used. FPGA interconnect also has the requirement of being "programmable". This is achieved with the front-end multiplexer located just before the signal driver.

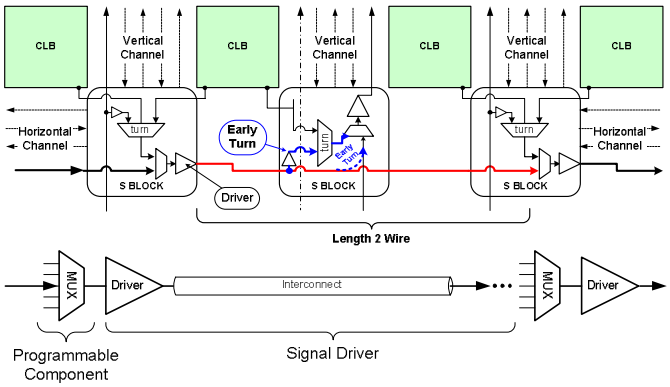



**Figure 3. FPGA interconnect and a simplified model.**

The fan-in of the programmable front-end multiplexers depend upon the precise FPGA architecture definition, but it can be anywhere between 2:1 and 64:1. There are several ways to implement multiplexers, as shown in Figure 4. The relative merits of each style are given in Table 1. We adopted the 2-level hybrid multiplexer, used in the Stratix II architecture [3], to improve delay. We also added the Stratix II fast input path, as shown in Figure 5, to optimize our driver designs for the fastest possible performance through these paths. Unlike the Stratix work, our multiplexers are all built using full CMOS pass gates with minimum-sized NMOS and PMOS transistors. This maintained full signal swing and improved delay.

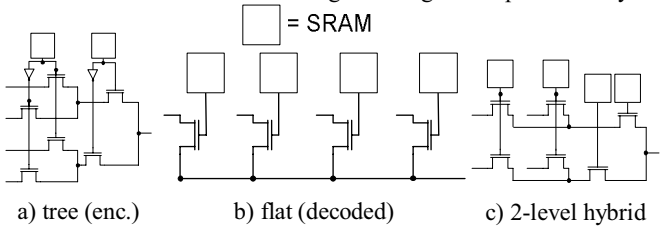



**Figure 4. Multiplexer design styles.**

| Feature \ Style | a) tree | b) flat | c) 2-level |
|---|---|---|---|
| delay levels | $\lceil \log_2 N \rceil$ | 1 | 2 |
| internal load (capacitance) | $\lceil \log_2 N \rceil$ (distributed) | N-1 (lumped) | $2\sqrt{N}$ |
| config. bits | $\lceil \log_2 N \rceil$ | N | $2\sqrt{N}$ |
| pass gates | $2N-2$ | N | $\left(N+\sqrt{N}\right)$ |

**Table 1. Comparison of multiplexer design styles (N = fan-in of mux).**

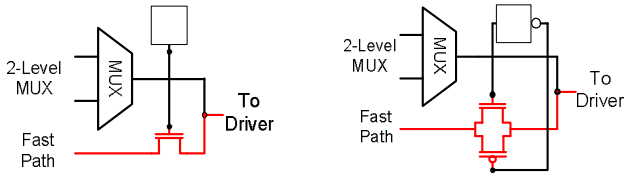



**Figure 5. "Fast Path" input is added to the 2-level multiplexer design.**

The design of interconnect optimization using buffers is well studied in the ASIC domain. Studies on wire sizing, power optimization and area reduction performed in [4-11] are achieved using closed form expressions to model buffers. Most work uses Elmore-based delay models which uses effective resistances. This approach is known to have modeling errors and does not accurately model the effects of slew rates [12] or signals with reduced swing. Rather than cope with

Elmore modeling inaccuracies, we chose to exploit the structured FPGA environment to reduce the problem space associated with FPGA interconnect design, and instead focus on more accurate HSPICE-based delay models. This also enables relatively simple future extensions of this work, such as accurate power-optimized design, which would not be possible with the Elmore approach.

Prior work in FPGA circuit design also uses HSPICE to model the circuits [1, 13-16], but much of this was based on the use of tristate buffers. Past work [2] develops an Elmore interconnect delay model into automated FPGA interconnect design. In this paper we develop much more accurate delay models using HSPICE. This gives us greater confidence in the results, and allows one to explore more general circuit design techniques such as low-swing interconnect by simply replacing the templates used within the HSPICE decks.

The **FPGA interconnect design problem** can be stated as follows. **Given** metal wire RC properties and a target total physical wirelength L, **find** the optimum number of inverters N, sizes of each inverter $B_0$ .. $B_{N-1}$, and amount of wire between each inverter $L_0$..$L_{N-1}$ to result in minimal signal propagation delay from the start of the interconnect wire to its end. A diagram and description of these parameters appear in Figure 6 and Table 2. To make this problem FPGA-specific, a front-end multiplexer must be included in this circuit to provide programmability. Also, the signal arrival time at interior points to the wire do matter in an FPGA, but we do not explicitly optimize for that version of the problem here.

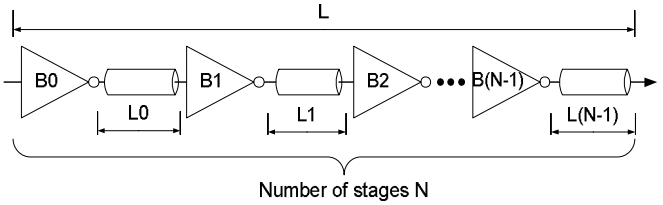



**Figure 6. General buffer sizing and spacing problem for FPGAs.**

| Parameter | Symbol | Description |
|---|---|---|
| total wirelength | L | Length of interconnect. This is the physical distance between multiplexers in an architecture (typically in mm). |
| number of driver stages | N | Number of inverter-wire fragments which make up the total interconnect wire, including the programmable component as the first stage. |
| buffer sizing | B*i* | Size of the buffer *i*, normalized to a minimum sized buffer. |
| buffer spacing | L*i* | Length of wire following buffer *i* (typically in mm, but may be expressed as a pertcentage of L). |

**Table 2. Design parameters for buffer sizing and spacing problem.**

## III. INTERCONNECT DESIGN FRAMEWORK

This section presents three different approaches for determining the number and size of inverters used to get low delay results from an FPGA interconnect wire.

The methods all assume that wire RC has been predetermined by choosing a predominant metal layer, wire width and spacing for a given technology node. In all cases when HSPICE was used to determine delay, an *identical* circuit was placed before and after the measured circuit for input shaping and load as shown in Figure 7.

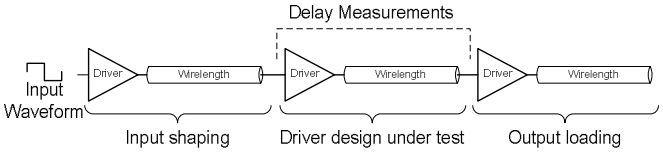



**Figure 7. Testbench used for measuring delay of one stage.**

### *A. Method A: Lumped Design (HSPICE based)*

We start with a simple transistor-level design method. The circuit being designed consists of a fast 2-input multiplexer, followed by a chain of tapered inverters, ending with the full length of the interconnect wire. This method determines the number and size of each inverter in the taper for best delay. Since all of the inverters are located at the beginning of the wire, this is called a *lumped design*.

The design method consists of the following nested loops:

1. Sweep the number of inverters (N) over a reasonable range of values (outer loop);
2. Sweep the final inverter size (inner loop);
3. Calculate the tapering factor (assuming geometrically increasing sizes and a minimum inverter size of 1.0x);
4. Determine circuit delay to endpoint using HSPICE.

HSPICE is used within the innermost loop to accurately determine the delay of each design. After this procedure, the circuit design found with the best delay is found, normalized (per mm) [17], and reported in Table 3 for several physical wirelengths.

| Wirelength (mm) | Number of stages (N) | Inverter sizes (x min. size) | HSPICE delay (ps/mm) |
|---|---|---|---|
| **Lumped design results (180nm, 1x spacing, 1x width)** | | | |
| 0.5 | 2 | 3.7, 14 | 408 |
| 1.0 | 3 | 4.0, 10, 30 | 260 |
| 2.0 | 3 | 4.0, 9.0, 35 | 192 |
| 3.0 | 3 | 3.3, 11, 37 | 184 |
| 4.0 | 4 | 2.7, 7.1, 19, 50 | 191 |
| **Lumped design results (90nm, 2x spacing, 2x width)** | | | |
| 2.0 | 3 | 4.0, 16, 65 | 115 |
| 3.0 | 5 | 2.6, 6.6, 17, 43, 110 | 115 |
| 4.0 | 5 | 2.6, 6.8, 18, 46, 120 | 125 |

**Table 3. Solutions found for Method A lumped design approach. The reported HSPICE delay includes a 2:1 front-end multiplexer.**

### *B. Method B: Distributed Design Using Nested Sweeps (Elmore based)*

In this section, we provide another simple transistor-level design method based on the Elmore delay model. The circuit being designed consists of the following elements connected in series: a minimum size inverter, a percentage $L_0$ of the total wirelength, an inverter of size $B_1$, a percentage $L_1$ of the total wirelength, an inverter of size $B_2$, and the remaining wire. The approach determines the size of inverters $B_1$ and $B_2$ and their location along the wire for best delay.

The design method consists of the following nested loops:

1. Fix the size of first inverter $B_0 = 1.0$;
2. Sweep $L_0$ from 0..100% of total wirelength (outer loop);
3. Sweep $L_1$ from 0..(100-$L_0$)% of total wirelength;
4. Calculate $L_2$, the remaining wire, as $1.0 - L_0 - L_1$
5. Sweep the inverter size $B_1$;
6. Sweep the inverter size $B_2$ (innermost loop);
7. Determine circuit delay using Elmore model.

Due to the extensive nesting of loops, Elmore delay is used to quickly evaluate delay at each design point and find the best design in terms of delay-per-mm. The best circuit design found for several physical

wirelengths is reported in Table 4. Final delay results are calculated for these designs using HSPICE. Also reported are the best circuit designs found according to the area-delay product.[2]

For comparison, the best lumped delay results were also obtained by repeating the same approach but fixing $L_0$ and $L_1$ in steps 2 and 3 to 0%. The distributed delays are better than lumped only after the physical interconnect length exceeds 2.5mm.

It is noteworthy that, for all interconnect lengths, the optimum delay was found with no wire located between the first and second inverters. As the lengths get longer, the approach tends to place the third buffer towards the middle of the wire.

Compared to Table 3, the delays in Table 4 are smaller because the front-end multiplexer was excluded throughout Method B. This was done because it was difficult to explore the many possible front-end circuit options (e.g., NMOS pass transistor with optional level-restorer, CMOS pass gate, etc.) using the Elmore approach (each would require a new manually-tuned model, making this more labour-intensive than automated). This limitation was the first factor motivating the development of Method C.

The results in Table 4 suggest that interconnect lengths greater than 4mm should have more than 3 inverters. Method B is limited to 3 inverters because it is meant to be relatively quick to compute with only 4 nested loops. Each additional inverter adds two more nested loops, one for buffersize and one for wire position, leading to much slower searches. Hence, the practical need to add more inverters motivated us to find a faster way to determine buffer locations. This was the second factor motivating the development of Method C, which is described in the next subsection.

| Wire-length (mm) | Best distrib. inverter locations (%) | Best distrib. inverter sizes | Distrib. HSPICE delay (ps/mm) | Lumped HSPICE delay (ps/mm) | Perf. diff. |
|---|---|---|---|---|---|
| **Delay-driven results (180nm, 1x spacing, 1x width)** | | | | | |
| 1.0 | 0.00 0.00 1.00 | 1 4 21 | 185 | 185 | 0% |
| 2.0 | 0.00 0.00 1.00 | 1 5 36 | 153 | 153 | 0% |
| 2.5 | 0.00 0.15 0.85 | 1 7 38 | 152 | 153 | 1% |
| 3.0 | 0.00 0.25 0.75 | 1 8 39 | 151 | 157 | 4% |
| 4.0 | 0.00 0.35 0.65 | 1 10 39 | 153 | 172 | 11% |
| 8.0 | 0.00 0.45 0.55 | 1 14 36 | 187 | 262 | 29% |
| 16 | 0.00 0.50 0.50 | 1 22 36 | 284 | 475 | 40% |
| **Area*Delay-driven results (180nm, 1x spacing, 1x width)** | | | | | |
| 1.0 | 0.00 0.00 1.00 | 1 3 13 | 195 | 195 | 0% |
| 2.0 | 0.00 0.00 1.00 | 1 3 16 | 167 | 167 | 0% |
| 2.5 | 0.00 0.00 1.00 | 1 3 17 | 169 | 169 | 0% |
| 3.0 | 0.00 0.40 0.60 | 1 7 14 | 168 | 176 | 5% |
| 4.0 | 0.00 0.45 0.55 | 1 8 14 | 169 | 201 | 16% |
| 8.0 | 0.00 0.50 0.50 | 1 9 12 | 211 | 307 | 31% |
| 16 | 0.00 0.50 0.50 | 1 8 10 | 325 | 553 | 41% |

**Table 4. Method B distributed buffer designs with N = 3 inverters using Elmore delay model. (Note: delay of front-end multiplexer is excluded)**

### *C. Method C: Distributed Design Using Concatenation (HSPICE based)*

This subsection presents a fast, accurate, HSPICE-based method for determining the best repeater design for a given target interconnect wirelength.

[2] Area was calculated according to the VPR area model, which computes the area of wide transistors in units of "# of min.-size transistor areas".

The method works by first *precharacterizing* two types of circuits using HSPICE, the multiplexer stage and distributed drive stage shown in Figure 8, into lookup tables. The lookup tables are indexed by the length of wire located after an inverter to provide delay and optimal buffersize. Next, total interconnect delay is quickly determined by adding values from the lookup tables and the lowest-delay solution is chosen. Based on results from Method B, previous work [9], and to simplify the design space , it is reasonable to assume that all of the distributed drive stages contain the same buffersize $B_1$ and the same physical wirelength $L_1$ to achieve near-minimum delay.

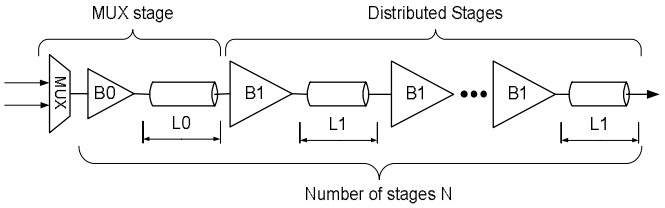
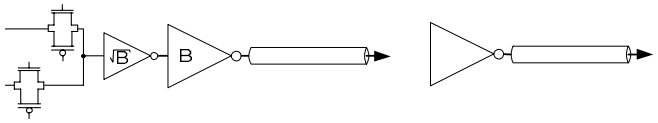



**Figure 8. Distributed buffer sizing and spacing problem, simplified.**

Several different full-swing circuit designs were tested with HSPICE for both the multiplexer stage and the distributed stage. Details of several transistor-level designs considered are provided in [18] but omitted here for brevity. The fastest results were obtained with the circuits shown in Figure 9. The 2 adjacent inverters in the mux stage were consistent with Method B results. The 2:1 multiplexer is built with minimum size transistors for both PMOS and NMOS.




a) multiplexed stage (creates a fast input) b) distributed drive stage

**Figure 9. Transistor-level design of mux and distributed drive stages.**

The first step in Method C consists of the following nested loops to perform **design precharacterization**:

1. For each mux or distributed drive circuit design;
2. Sweep the physical wirelength from 0 to 2mm (outer loop);
3. Sweep final buffersize B (innermost loop);
4. Determine circuit delay using HSPICE.

The results computed by this first step can be summarized by two 2D arrays, mux_stage_delay[] and distrib_stage_delay[], which are indexed by physical wire length and buffersize.

The second step in Method C consists of the following nested loops to do **circuit construction**:

1. Sweep N, the number of stages (outermost loop);
2. Sweep $L_0$ from 0..100% of total wirelength (inner loop);
3. Calculate $L_1$, the wire per distrib. stage as $(1.0 - L_0) / N$;
4. Calculate circuit delay by adding delay lookup tables: mux_stage_delay[$L_0$] + (N-1) * distrib_stage_delay[$L_1$].

For fast runtimes, the precharacterization step is limited to 2 nested loops. This step produces a lookup table which can be indexed by [wirelength][buffersize] to accurately determine delay. This data can actually be compressed into a one-dimensional array, indexed only by [wirelength]. To do this one can find the buffersize that obtains minimum delay and store both delay and corresponding buffersize at the corresponding wirelength index. We noticed that the optimal buffersizes used for the distributed stage were rather large, so we relaxed the delay target to 10% above minimum delay and found the corresponding buffersize. This nearly cut buffer sizes in half, from roughly 50x to 30x for 180nm, but did not significantly increase

overall delay. The optimal buffersizes used for the mux stage were already small enough, and there is only one mux stage, so this same area-saving technique was not applied there. Once computed, this lookup table data can potentially be embedded into an architecture file for VPR, and VPR can run the circuit construction step internally to very quickly determine delay and best buffer sizes and locations for each interconnect wirelength.

The best delay-per-mm data obtained from the precharacterization step are summarized in Table 5. Although this data is not directly needed by Method C, it provides two useful reference points. First, the distributed drive stage data provides the best ASIC repeater delay for the assumed metal layer, wire width, and wire spacing. Second, the mux stage data indicates the best FPGA interconnect delay obtained by cascading *only* mux stages. Table 5 also shows multiplexed signal delays are ~2x ASIC delays. The use of distributed stages after a mux stage can help improve FPGA delay, but not beyond the lower bound formed by the ASIC repeater delay.

| Process | FPGA interconnect stage | Delay (ps/mm) 1x spacing, 1x width | Delay (ps/mm) 2x spacing, 2x width |
|---|---|---|---|
| 180nm | multiplexed | 207 | 138 |
| | distrib. drive* | 108 | 69 |
| 90nm | multiplexed | 199 | 131 |
| | distrib. drive* | 91 | 58 |

* also corresponds to best possible ASIC repeater delay (full-swing)

**Table 5. Best delay-per-mm for different stages and wire models extracted from precharacterization data of Method C.**

| Wire-length (mm) | # of Stages (N) | 1 Multiplexed stage | | N-1 Distributed stages | | Concat. delay (ps/mm) |
|---|---|---|---|---|---|---|
| | | Driver size B0 (x min.) | Length L0 (mm) | Driver size B1 (x min.) | Length L1 (mm) | |
| **Distributed design results (180nm, 1x spacing, 1x width)** | | | | | | |
| 0.5 | 2 | 3.0 | 0.05 | 14 | 0.45 | 414 |
| 1.0 | 2 | 6.0 | 0.15 | 22 | 0.85 | 266 |
| 2.0 | 3 | 6.0 | 0.15 | 22 | 0.93 | 196 |
| 3.0 | 4 | 11 | 0.36 | 26 | 0.88 | 170 |
| 4.0 | 4 | 14 | 0.60 | 27 | 1.13 | 157 |
| **Distributed design results (90nm, 2x spacing, 2x width)** | | | | | | |
| 2.0 | 4 | 12 | 0.15 | 43 | 0.62 | 103 |
| 3.0 | 5 | 17 | 0.26 | 47 | 0.69 | 90 |
| 4.0 | 6 | 19 | 0.30 | 48 | 0.74 | 84 |

**Table 6. Method C distributed buffer designs with 1 mux stage and N-1 identical driver stages using HSPICE precharacterization data. The front-end mux is included in the total delay (Method C concatenation).**

Using Method C, we designed several transistor-level circuits with excellent delay performance for several different interconnect wirelengths. These designs are reported in Table 6. For each design in this table, we generated a full HSPICE deck and verified that the delay estimate calculated by the circuit construction step is within 4% error of actual HSPICE delay [18]. Recall that short wirelengths ≤ 2mm obtained best performance from a lumped driver. Method A's thorough HSPICE sweeps of *lumped designs* is not much better than the *distributed designs* of Method C. Yet, distributed designs deliver signals earlier to wire midpoints. For lengths > 2mm, Method C is superior to Method A due to the strong need for distributed buffering.

From Table 6, one might notice that 90nm has twice the performance of 180nm. Notice that the 90nm design is using twice the minimum metal width and spacing but 180nm is using minimum, *i.e.,* the wires are physically the same width and distance apart, hence have similar RC time constants per unit length. The key difference is improved transistor performance, which leads to *more drive stages* and *relatively wider transistors* (but similar physical width) for the same total interconnect length. However, one must question whether a 4mm wire in a 180nm FPGA must remain 4mm when the architecture is redesigned for 90nm. Intuitively, since transistor dimensions are cut in half, a comparable wire only needs to be 2mm in 90nm.[3] The next subsection will examine the delay of interconnect from a top-down approach by planning the overall physical length needed for high-performance interconnect wires.

### *D. Putting It Together: Multiplexing Interval*

We applied Method C to a variety of interconnect lengths from 1mm to 10mm and plotted the delay-per-mm results in Figure 10. Since the x-axis represents the planned total physical length of the interconnect wire, we call this the *multiplexing interval*. This can also be viewed as the distance between "programmable" points in the wire. For reference, we also plotted the best ASIC signal velocity (which is independent of total wirelength). Data for both 1x/1x and 2x/2x wire width/spacing is provided.

Referring to Figure 10, we see the delay performance of a 4mm wire in 1x/1x/180nm is ~155ps/mm. If the device is rescaled to 90nm, the wire becomes 2mm long and delay improves by only 10% to ~140ps/mm. Increasing width/spacing of the 90nm wire improves delay to ~105ps/mm. Alternatively, one might plan to include *longer* architectural wires in the FPGA device at 90nm. At min. width/spacing, delay improves to ~115ps/mm at 4mm or ~110ps/mm at 10mm. Thus, it is possible to *improve* signal speed by planning to lengthen the architectural wirelength. However, a lower bound is still formed by the ASIC delay.

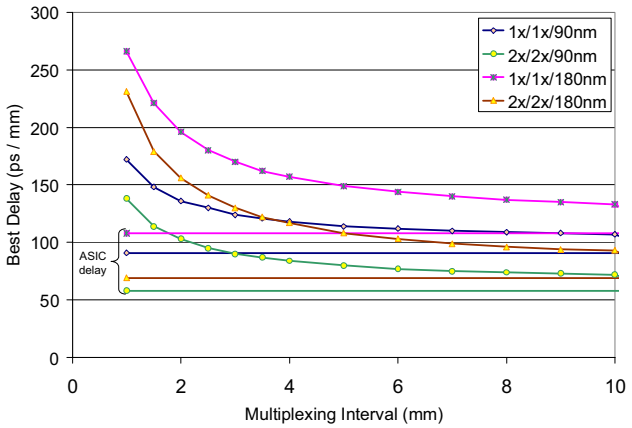



**Figure 10. Determining a multiplexing interval for FPGA interconnect.**

### *E. Putting It Together: Path Delay Profile*

Method C optimization is driven by HSPICE delay to the final endpoint. However, in the introduction of this paper it was argued that we must also consider signal arrival time at all possible interior points along the wire since it is not known in advance whether the critical path will use the entire physical length of the wire.

Path delay profiles for 2mm and 3mm interconnect are plotted in Figure 11 for the best lumped designs and for several different distributed designs (with N=2, 3, or 4 total stages). From this metric, we see that several interior points within the first 1mm of a 2mm interconnect arrive *earlier* than the lumped design, but the points past 1mm arrive slightly *later*. At a distance of 0.7mm, distributed is 50ps (15%) faster and at 2.0mm it is 40ps (10%) slower. Overall, it is not

[3] This assumes no extra "stuff" is added to a CLB layout tile (no extra logic, no additional routing tracks), which may not be true!

clear whether a distributed or lumped buffer design would be faster for 2mm interconnect after a circuit is placed and routed. This will be investigated in the next section, which shows that distributed is better.

For 3mm interconnect, we notice that distributed buffering is *always* faster than lumped. Hence, at some point, distributed design is always a better choice. Furthermore, the path delay profile demonstrates that significant delay savings can be obtained by more accurately modeling delay to the interior points of a wire. At 1.5mm, the difference is 115ps (465ps – 350ps, or 25%), which is quite large.

VPR assumes that all points along the wire receive the signal at the same time. However, for long interconnect wires, it becomes very important to model delay quite accurately during place and route. This provides motivation for increased modeling accuracy in VPR, which is also pursued in the next section.

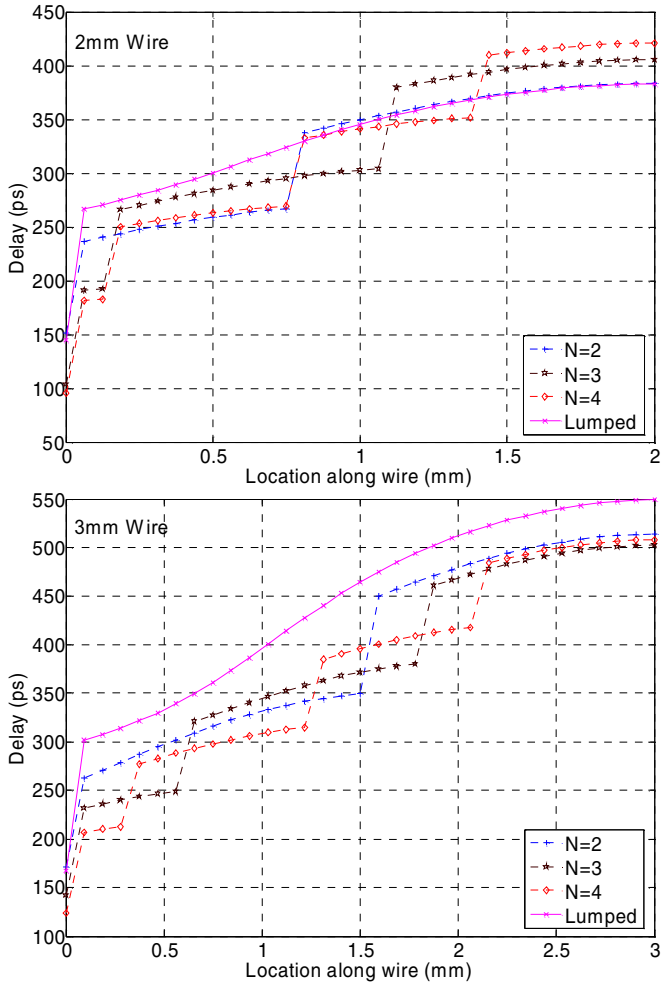



**Figure 11. Path delay profiles show advantages of distributed buffering.**

## IV. Delay Modeling in Place and Route

The previous section demonstrated the need to distribute buffers along the length of long interconnect wires. It also suggested that more accurate delay modeling is needed for long interconnect wires because early signal arrival can result in significant delay improvements. Clearly, proper modeling is not just needed for accurate timing analysis, but it is also essential for the router to make the best possible choices. In this section, we present the results of adding this more accurate delay modeling to VPR, which we call the ETM or *early turn model*, and use it to explore changes in router preferences. We also consider the performance impact of adding *fast input* paths to multiplexers to assist in accelerating straight-through connections.

### A. VPR Changes

Originally, VPR estimates delay to *all points* on a wire as delay to the halfway point using the formula ½ ($R_{wire}C_{wire}/2$). With ETM, interconnect wires are broken into wire fragments, where each fragment spans one CLB. This allows VPR to use its Elmore delay calculator to more accurately calculate interior point delays.

The circuit construction step of Method C was not directly embedded into VPR due to concerns about correct delay calibration of the internal Elmore calculations. In a research architectural exploration tool, such calibration is not overly important and could probably be omitted for most non-circuit-design research. However, for this paper, we considered the accuracy of modeling a different number of stages to be very important for capturing the true performance benefit of ETM and distributed buffering.

To calibrate, we created a test netlist of a long, straight connection and extracted delays in VPR from the start to every CLB along the way. We plotted this VPR-extracted path delay profile (PDP) and compared it to the HSPICE-computed PDP. The VPR architecture file models buffers with an equivalent $R_{out}$ for Elmore delay calculations: we initialized $R_{out}$ according to the size of the buffer, but then made minor manual adjustments (up to 20%) so the VPR-PDP matched the HSPICE-PDP as closely as possible. There is some mismatch because VPR must snap buffer locations to the array grid (placing it in either one CLB or the next, not halfway in between). Also, VPR adds a small additional C load according to the number of signal taps along the wire.

VPR must also include the full delay of large multiplexers in the interconnect. We model straight connections using a "fast path", or one input of the 2:1 multiplexer. All other turns must utilize a wide fan-in mux, or "slow path". We built several multiplexer sizes of 2-level hybrid multiplexers in HSPICE, extracted delays, and created a parameterized delay model. Wherever a wide fan-in multiplexer is used, the additional delay is calculated according to this model and added into the VPR routing graph.

Finally, the VPR router and timing analyzer were checked to ensure that incremental delays were being calculated correctly, and that the router was taking advantage of early turn delay data to make improved routing decisions.

### B. Results

The traditional 20 MCNC benchmark circuits were mapped into 4-input LUTs, packed into CLBs containing 8 BLEs, and placed once. Then, each placed benchmark was routed several times, each using different interconnect designs. The same channel width is used to route a benchmark across interconnect designs of the same logical wirelength. This was determined by first doing a binary search to find the minimum, then adding a fixed amount of additional wires (8 or 32 for L4 or L16, respectively) to relax the router and improve critical path results. For a particular logical wirelength, the maximum channel width needed across different buffering designs was then used for *all* final routings of that benchmark.

The normalized geometric average critical path delay is reported in Table 7 for several different interconnect lengths. The critical path delays are normalized according to the results obtained using the circuit design published in [1] and labeled *FPT04 Design* in the table.

The FPT04 circuit was designed only for 0.46mm physical wire lengths, so it is expected to be slower on longer wires.

The performance of several different target interconnect lengths are provided. The 0.5mm wirelength is assumed to correspond well to a logical wirelength of 4 CLBs (L4). For the 2mm and 3mm wirelengths, a logical wirelength of 16 CLBs (L16) is chosen. Since the MCNC circuits are quite small, they do not fully "exercise" or exploit very long wires properly. To compensate, the benchmark circuits were packed to utilize only 1 LUT per CLB for L16 logical wirelengths; this spread out the circuit onto a much larger array size. Although this does not truly represent a real FPGA architecture (which would have mixed wire lengths) or a large circuit (which would have naturally long paths), it highlights the benefits.

From the results in Table 7, the lumped design approach described here improves the FPT04 design of 0.5mm wires by 10%. Better delay modeling (ETM) and better routing decisions account for an additional 2% delay improvement. Addition of the "fast path" to straight L4 connections improves delay by another 7%, resulting in a total of 19% improvement to the FPT04 design.

| Design | Lumped | | | | | Distributed | |
|---|---|---|---|---|---|---|---|
| Wirelength * | FPT04 Design | Lumped | Lumped +Fast | Lumped +ETM | Lumped +ETM +Fast | Distributed +ETM | Distributed +ETM +Fast |
| 0.5mm (L4) | 1.0 (20ns) | 0.90 | 0.82 | 0.88 | 0.81 (16.2ns) | - | - |
| 2.0mm (L16) | 1.0 (31ns) | 0.73 | 0.70 | 0.69 | 0.65 | 0.67 | 0.63 (19.5ns) |
| 3.0mm (L16) | 1.0 (38ns) | 0.70 | 0.67 | 0.63 | 0.60 | 0.56 | 0.54 (20.5ns) |

* L4 packs CLBs until full, L16 packs only 1 LUT per CLB to spread out the circuit over a larger array (creates a need for long connections)

**Table 7. Normalized critical path delay results.**

The 2mm and 3mm wirelengths achieve even more significant performance improvements using the design approach described in this paper. First, the lumped design improves by 27-30%, the use of better delay modeling and routing decisions improves another 4-7%, and use of the "fast path" improves another 3-4%. Second, switching to distributed design helps *both* the 2mm and 3mm delays. *Even though the raw end-to-end delay performance of 2mm distributed interconnect is worse than lumped as seen in the PDP, there is a small 2% net improvement in delay due to the importance of early turns and using ETM.* The importance of distributed design and ETM are both magnified in the 3mm design. Overall, the 3mm wirelength is only 1ns slower (5%) than the 2mm wirelength! Also, the 3mm wirelength achieves an impressive 46% reduction in critical path delay compared to the FPT04 design.

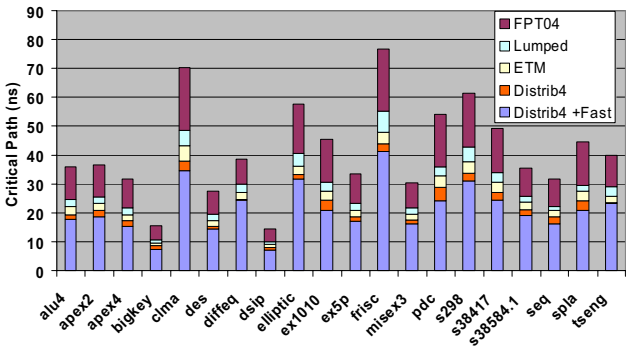



**Figure 12. Delay breakdown for a 3.0mm wire.**

A breakdown of delay performance on a circuit-by-circuit basis for the 3mm wirelength is provided in Figure 12.

The addition of distributed buffering and improved delay modeling affects decisions made by the router on when/where to turn. Based on this, we decided to investigate the impact of this by analyzing the frequency of turns in the routing solution. Figure 13 defines what we mean by "early turn" and "normal turn". Early turns occur at any switch block (except the last one) or to any CLB (except the last one). Normal turns occur at the last CLB, or in the last switch block when the signal changes direction. If the signal continues straight, that is a "straight thru".

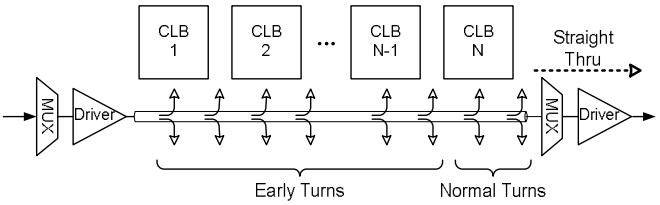
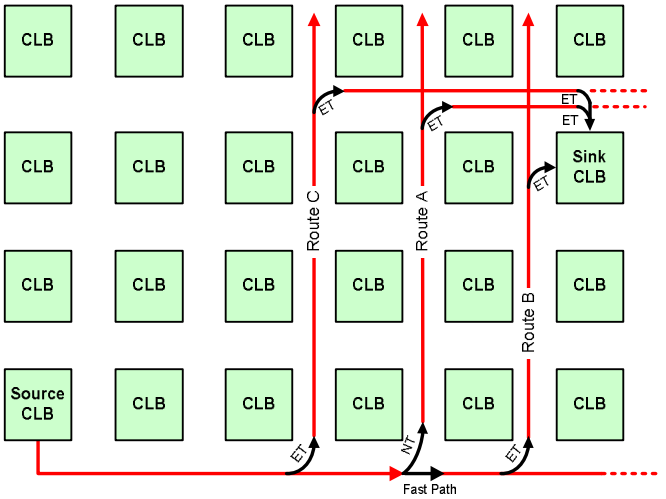



**Figure 13. Early turns and normal turns.**

Table 8 gives average turn data across the benchmark circuits. The addition of the multiplexer "fast path" and ETM does not significantly change the total number of turns. However, it does affect the distribution: early turns and straight thrus both increase slightly at the expense of normal turns. In the 2mm case, normal turns are reduced by 26% (from 6.9% of all turns to just 5.1%). This is a significant reduction in turns at the end of a wire. We suggest further study into this, but it appears that making turns at the end of a wire is becoming less important. This is a significant architectural result because it suggests less switching flexibility is needed in S blocks.

| Designs | Average Total Number of Turns | Average Early Turns | Average Normal Turns | Average Straight Thrus |
|---|---|---|---|---|
| Lumped 0.5mm | 7587 | 57.3% | **25.7%** | 16.0% |
| + Fast | 7591 | 57.3% | 24.2% | 17.5% |
| + ETM + Fast | 7698 | **59.2%** | 20.8% | **18.9%** |
| Lumped 2.0mm | 10978 | 87.6% | **6.9%** | 4.9% |
| + Fast | 10908 | **88.4%** | 6.4% | 4.5% |
| + ETM + Fast | 11057 | **88.4%** | 5.1% | **5.3%** |
| Lumped 3.0mm | 10983 | 87.5% | **6.9%** | 5.0% |
| + Fast | 10913 | **88.3%** | 6.5% | 4.5% |
| + ETM + Fast | 11073 | 88.2% | 5.2% | **5.5%** |

**Table 8. Effect of adding Fast Paths on turn counts.**




| Routing | No ETM No Fast | No ETM + Fast | + ETM + Fast |
|---|---|---|---|
| A - "Normal Turn" | 1 | 2 | 3 |
| B - "Straight Thru" | 1 | 1 | 1 |
| C - "All Early Turns" | 1 | 2 | 2 |

**Figure 14. Early turns alters rank of routing preferences.**

To help understand why turns at the end of a wire are less important with fast paths and ETM, we constructed a hypothetical case shown in Figure 14. Originally, VPR was unable to distinguish the performance of routes A, B, and C: all paths use 3 wires and have the same delay. With the fast path, Route B becomes preferred because it has lower delay along one turn. With ETM also enabled, Route A becomes less attractive than Route C because C initially appears to have lower delay during lowest-delay wavefront expansion. Although this helps explain the demotion of normal turns to early turns and straight thrus, it does not account for all cases. We think this will also mildly impact the frequency and type of hardwired turns suggested in [19].

## V. Conclusions

This paper has demonstrated three methods for solving the FPGA interconnect design problem. One method produces lumped designs with all inverters at one end, while two methods provide distributed design solutions. The last method is quite fast and accurate, achieving delay errors within 4% of HSPICE, because it is based upon precharacterizing a circuit with HSPICE. It is also fast enough to be embedded in an FPGA architectural exploration tool.

Using the best interconnect designs found, we demonstrated an average critical path delay reduction of 19% using 0.5mm wires (logical architectural length 4) compared to previous work. With longer 2mm and 3mm wires, the improvement was even more dramatic at 27% and 46%, respectively. For these longer wires, most of the benefit came from optimizing the buffers for the specified wirelength, demonstrating the importance of being aware of the physical design constraints. Even so, significant benefits can be attributed to improved delay modeling and early turn use in VPR, as well as improvements due to the superiority of distributed buffer design. With all of these changes, we witnessed a ~25% reduction in the fraction of "normal turns" that occur at the end of a wire. This may have impact on architectural decisions and should be studied in future work.

This work has focused on obtaining the best end-to-end circuit delay performance. Future work could modify this to optimize for early turn performance as well – we briefly investigated this by integrating the area under the PDP curve and found it to be a useful optimization metric. As well, we noticed that the design space can sometimes be relatively tolerant to small changes in buffer location and buffer sizes. This means there is significant room to also address additional optimization goals such as minimum-power switching as well.

## References

[1] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and Single-Driver Wiring in FPGA Interconnect," in *Int'l Conference on Field-Programmable Technology*, Dec. 2004.

[2] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong, "Performance Benefits of Monolithically Stacked 3D-FPGA," in *Int'l Symp. on FPGAs*, pp. 113-122, Feb. 2006.

[3] D. Lewis et al, "The Stratix II Logic and Routing Architecture," in *Int'l Symp. on FPGAs*, pp. 14-20, Feb. 2005.

[4] V. Adler and E. G. Friedman, "Repeater Insertion to Reduce Delay and Power in RC Tree Structures," in *Conference on Signals, Systems & Computers*, pp. 749-752, Nov. 1997.

[5] V. Adler and E. G. Friedman, "Uniform Repeater Insertion in RC Trees," *IEEE Transactions on Circuits and Systems I*, vol. 47 no. 10, pp. 1515-1524, 2000.

[6] V. Adler and E. G. Friedman, "Repeater Design to Reduce Delay and Power in Resistive Interconnect," *IEEE Transactions on Circuits and Systems II*, vol. 45 no. 5, pp. 607-616, 1998.

[7] C. J. Alpert, J. Hu, S. S. Sapatnekar, and C. N. Sze, "Accurate Estimation of Global Buffer Delay Within a Floorplan," *IEEE Trans. on Computer-Aided Design*, vol. 25 no. 6, pp. 1140-1146, 2006.

[8] K. Banerjee and A. Mehrotra, "A Power-Optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs," *IEEE Transactions on Electron Devices*, vol. 49 no. 11, pp. 2001-2007, 2002.

[9] S. Dhar and M. A. Franklin, "Optimum Buffer Circuits for Driving Long Uniform Lines," *IEEE J. of Solid State Circuits*, vol. 26 no. 1, pp. 32-41, 1991.

[10] R. H. J. M. Otten, "Global Wires: Harmful?," in *Int'l Symp. on Physical Design*, pp. 104-109, April 1998.

[11] L. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," *IEEE Int'l Symp. on Circuits and Systems*, pp. 865-868, May 1990.

[12] N. Nassif, M. P. Desai, and D. H. Hall, "Robust Elmore Delay models Suitable for Full Chip Timing Verification of a 600MHz CMOS Microprocessor," *Design Automation Conference*, 1998.

[13] V. Betz and J. Rose, "Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect," in *IEEE Custom Integrated Circuits Conference*, pp. 171-174, May 1999.

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

[15] G. Lemieux and D. Lewis, *Design of Interconnection Networks for Programmable Logic*, Kluwer Academic Publishers, 2004.

[16] G. Lemieux and D. Lewis, "Circuit Design of Routing Switches," in *Int'l Symp. on FPGAs*, pp. 19-28, Feb. 2002.

[17] S. Sood, M. Greenstreet, R. Saleh, *A Novel Interleaved and Distributed FIFO for Source-synchronous Interconnect*, VLSI Design and Test Symposium, India, Aug. 2006.

[18] E. Lee, *Interconnect Driver Design for Long Wires in Field-Programmable Gate Arrays*, Masters thesis, Dept. of ECE, University of British Columbia, June 2006.

[19] S. Sivaswamy, G. Wang, C. Ababei, K. Bazargan, R. Kastner, and E. Bozorgzadeh, "HARP: Hard-wired Routing Pattern FPGAS," in *Int'l Symp. on FPGAs*, pp. 21-29, Feb. 2005.