Journal of VLSI Signal Processing 2007

DOI: 10.1007/s11265-007-0141-y

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

EDMUND LEE, GUY LEMIEUX AND SHAHRIAR MIRABBASI
University of British Columbia, Vancouver, Canada

Received: 15 April 2007; Revised: 31 July 2007; Accepted: 16 August 2007

Abstract. Each new semiconductor technology node brings smaller, faster transistors and smaller, slower wires. In particular, long interconnect wires in modern FPGAs now require rebuffering at interior points in the wire. This paper presents a framework for designing and evaluating long, buffered interconnect wires in FPGAs with near-optimal delay performance using HSPICE-derived delays. 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. While traditional hand-calculations used for ideal repeater placement can be used, they are not very accurate and ignore practical constraints such as the overhead effects of front-end multiplexing and driving logic, "finite" wire length, and a discrete number of repeaters. 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 0.5, 2, and 3 mm wire lengths in 180 nm 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 0.5 mm wires and by up to 46% for 3mm wires over previous designs.

Keywords: FPGA, FPGA interconnect, interconnect design, routing design, computer-aided design

1. 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 transistors become 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. Traditional hand calculations based on Elmore delays can be used to determine the number of repeaters, their size and separation distance, but they are only approximations based on certain assumptions: the wire length is infinite and every repeater must be identical and evenly spaced, including the first driver. As a result, these hand calculations ignore the practical need to include the effects of any front-end logic, and the requirement for an integral number of repeaters. Since real, finite-length wires typically need only one to ten repeaters, these effects can be very prominent. Also, the front-end logic must present a minimal load on preceding logic, so the assumption of using a very wide inverter at the beginning is impractical; the effect of any multiplexing and the initial inverter chain must be included. Finally, the Elmore hand calculations usually do not include any area constraints, so the calculated delay-optimal buffer sizes are usually far too large to be practical (100 to 200 times the minimum buffer size instead of 10 to 50 times).

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. One work, [2], employs the traditional hand calculations but doesn't improve upon the assumptions. 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" (PDP), shown in Fig. 1, is also used. The figure highlights why distributed design can be considered faster than lumping buffers at one end when end-to-end performance is similar—the signal arrives earlier at wire positions before the halfway point. This new method is used to design interconnect for 0.5, 2.0, and 3.0 mm wire lengths in 180 nm technology and demonstrate the efficacy of the approach. Even in an "older" technology node such as 180 nm, which was selected due to the availability of previously published FPGA delay results and previously characterized FPGA architectural delays, it is worth noting that this approach has considerable influence on results. Its significance grows as we scale further.

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.

2. Background and Problem Formulation

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

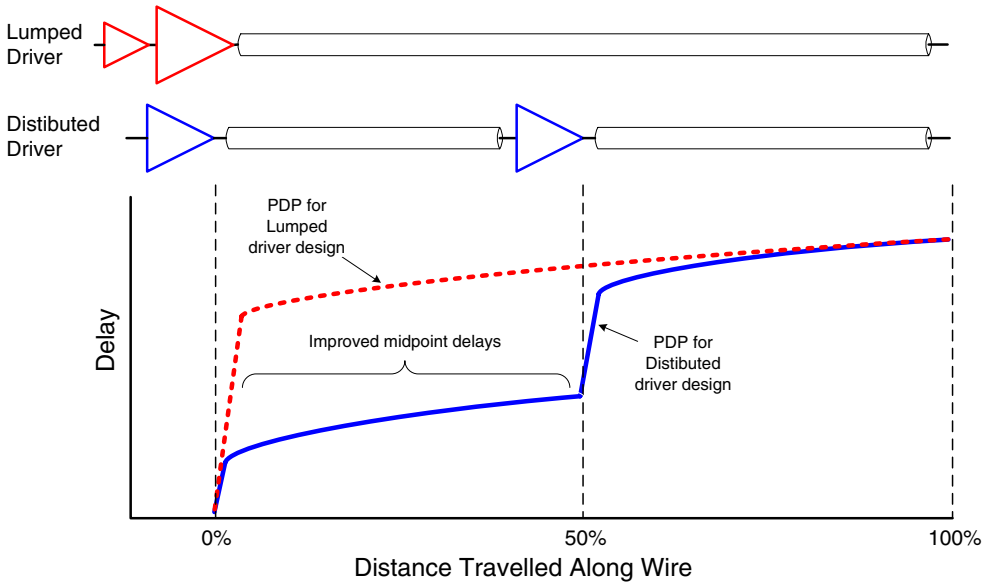


Figure 1. These path delay profiles suggest distributed buffering is faster because several interior points along the wire receive the signal earlier.

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

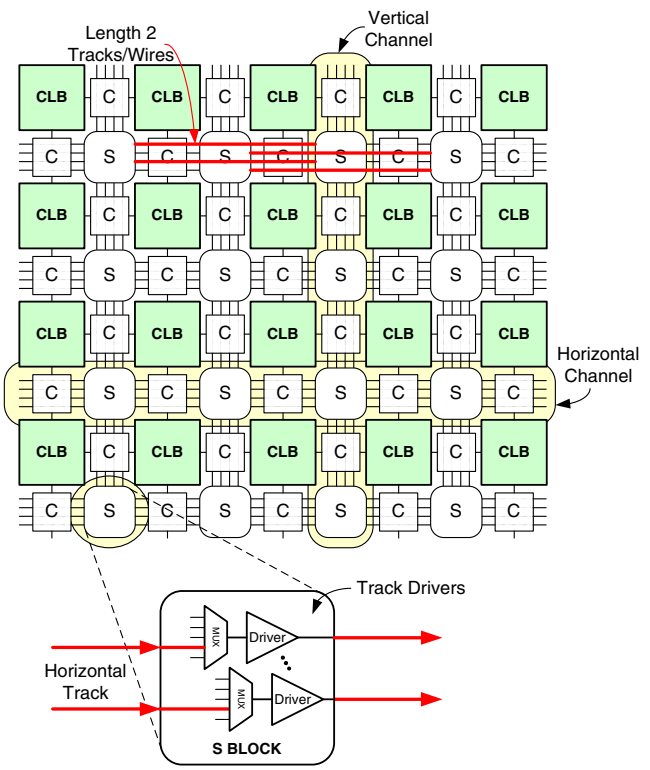


Figure 2. FPGA architecture and switch block detail.

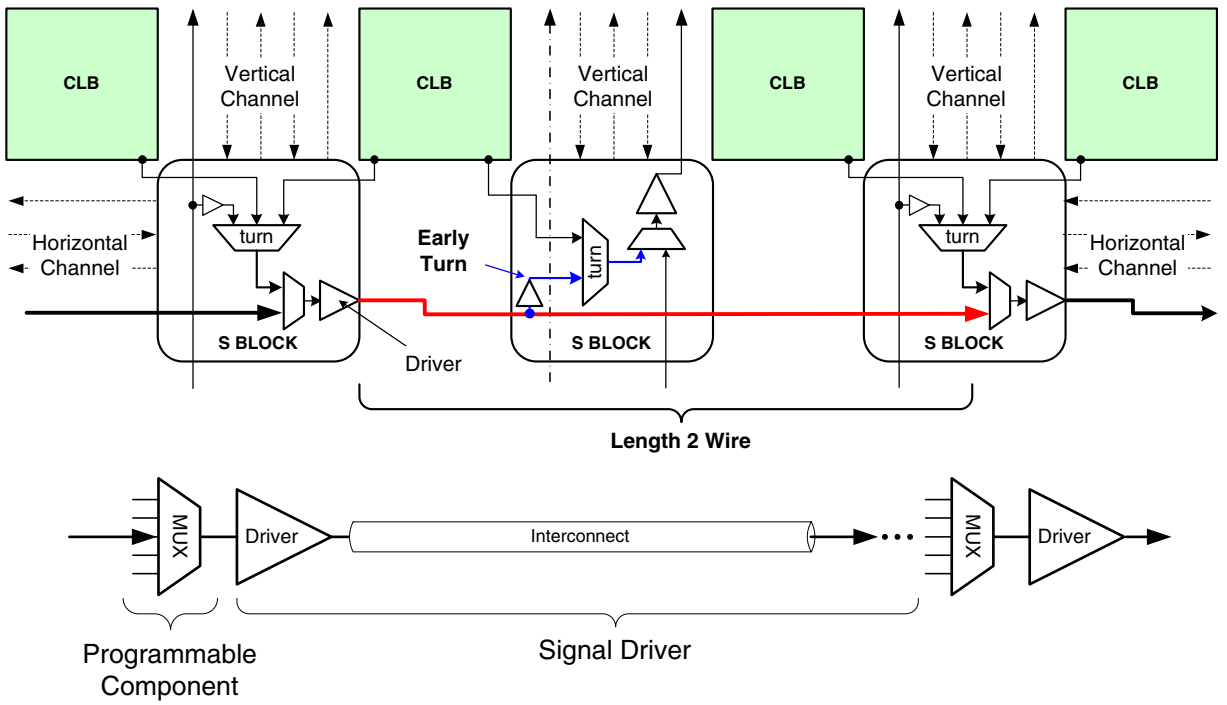


Figure 3. FPGA interconnect and a simplified model.

achieved with the front-end multiplexer located just before the signal driver.

The fan-in of the programmable front-end multiplexers depends 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 Fig. 4. The relative merits of each style are given in Table 1. We adopted the two-level hybrid multiplexer, used in the Stratix II architecture to improve delay [3]. We also added the Stratix II fast input path, as shown in Fig. 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.

Interconnect optimization design 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

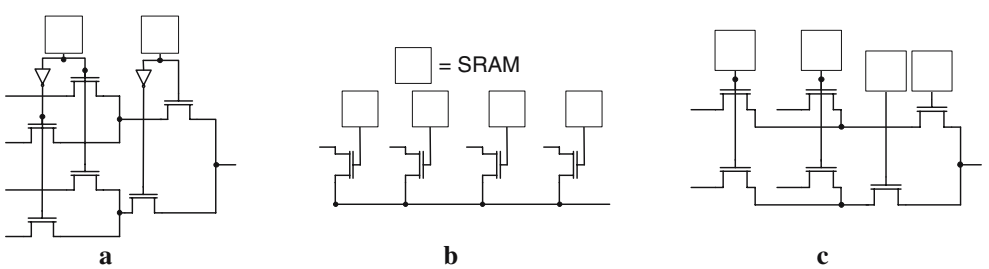


Figure 4. Multiplexer design styles. **a** Tree (encoded), **b** flat (decoded), **c** two-level hybrid.

Table 1. Comparison of multiplexer design styles (N is the fan-in of the multiplexer).

	Style		
Property	Tree	Flat	2-level
Delay levels	$[\log_2 N]$	1	2
Internal load (capacitance)	$[\log_2 N]$ (distributed)	$N-1$ (lumped)	$2\sqrt{N}$
Configuration bits	$[\log_2 N]$	N	$2\sqrt{N}$
Pass gates	$2N-2$	N	$N+\sqrt{N}$

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 buffer spacing and sizing 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 Fig. 6 and Table 2. To make this problem FPGA-specific, the delay influence of a front-end multiplexer must be included in this circuit to provide programmability. If the inverter B_0 requires rather large transistors, then additional tapered inverters will be needed between the multiplexer and B_0. However, we do not need to model this explicitly, since allowing L_0 to be 0 will produce the same effect. Also, the signal arrival time at interior points of the wire do matter in an FPGA, but we do not explicitly optimize for that version of the problem here.

3. 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, and 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 loading, as shown in Fig. 7.

3.1. Width and Spacing: Wire RC Characterization

Interconnect wire RC characteristics depend upon two values: fixed process parameters and variable design features. The fixed parameters consist of things like material dielectrics between wires and

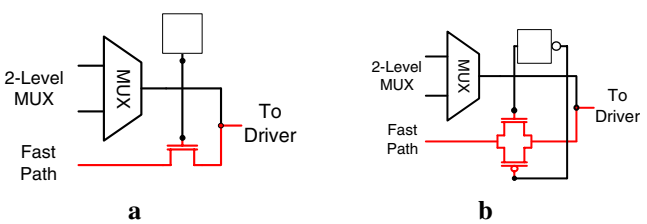


Figure 5. "Fast Path" input is added to the two-level multiplexer design. **a** NMOS pass transistor, **b** CMOS pass gate.

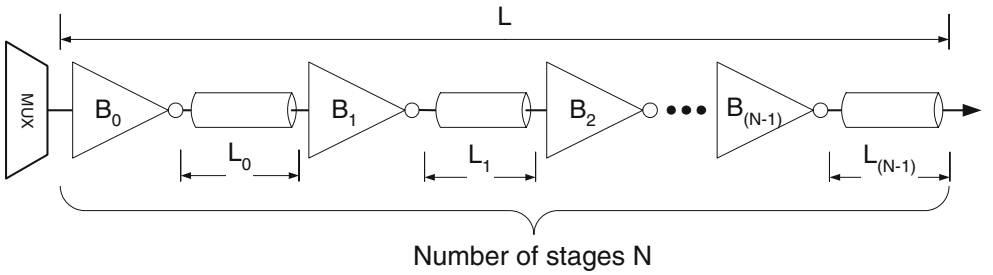


Figure 6. FPGA interconnect buffer spacing and sizing problem.

metal resistivity, which are defined by the technology process. Since the 180 nm technology node, the most significant improvements have been the shift from aluminum to copper (lower resistance) and development of low-k dielectric materials (lower capacitive coupling). These technology changes are limited in the amount of improvements available (e.g., it is difficult to improve upon copper as a conductor).

Fortunately, designers can influence interconnect RC by controlling design features such as wire spacing, wire width, and metal layer level. The effect of these features on capacitance can be seen in Fig. 8. Increasing the space between wires lowers the capacitive coupling between them. Widening the wire lowers resistance, which is the dominant effect, and increases plate capacitance between layers, which is a milder secondary effect due to the dominance of coupling capacitance. The choice of metal layer also has an effect: for manufacturability reasons, higher levels of metal are often thicker (larger coupling capacitance), have a larger minimum width and spacing than the lower layers, and may have different dielectric materials between the wires and between the layers above and below.

For this work, we assume interconnect is located in the metal 3 or metal 4 layers, since lower levels are usually occupied with local routing and higher levels are needed for power and clock distribution. A series of characterization experiments were conducted to determine suitable spacing and width of wires.

The effects of spacing on delay for a 4mm length of wire driven by a 20-times-minimum-size buffer are shown in Fig. 9. Delay results are expressed as multiples of a fanout-of-4 inverter delay (FO4) for both 90 and 180 nm technologies. As shown in the figure, a roughly 40% reduction in delay is possible by spreading wires out to four times the minimum spacing. The most significant drop occurs during the initial spreading, from 1× to 2× minimum spacing. To limit area inflation, we chose to use 2× spacing for all wires.

Starting from the 2× spacing result, the effect of width on delay is shown in Fig. 10 for the same 4 mm wire and 20× buffer. The figure shows that a compounded 35–40% delay reduction is possible through widening the wire. Again, most of this improvement comes from the initial doubling of the wire width, so we chose to use 2× widths for all wires.

Overall, wire RC delay is reduced by roughly 50% by the use of 2× spacing and 2× width. Unless otherwise stated, these are the spacing and width results that will be assumed throughout this work.

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

Parameter	Symbol	Description
Total wirelength	L	Length of interconnect. This is the physical distance between multiplexers in an architecture (typically in millimeters)
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. A length of 0 is allowed. (Typically in mm, but may also be expressed as a percentage of L)

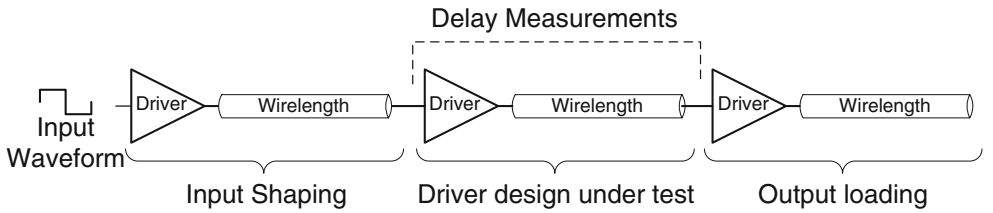


Figure 7. Testbench used for measuring delay of one stage.

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

We start with a simple transistor-level design method. The circuit being designed consists of a fast two-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:

```
for N = 1 to 4 // number of inverters (over a reasonable range)
   for B = 1 to 32 // final inverter size (can be fractional step size)
      calculate tapering factor from B and N
         // assuming geom. increasing sizes and a min. size of 1x
      determine circuit delay to endpoint using HSPICE
   endfor
endfor
```

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

3.3. *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 position along the wire for best delay.

The design method consists of the following nested loops:

```
B0 = 1.0  // fix the size of first inverter
for L0 = 0 to 100% // percent of total wirelength
   for L1 = 0 to 100%-L0
      L2 = 100% - L0 - L1  // the remaining wirelength
      for B1 = 1 to 64    // second inverter size (can be fractional)
         for B2 = 1 to 64 // third inverter size (can be fractional)
            determine circuit delay to endpoint using Elmore model
         endfor
      endfor
   endfor
endfor
```

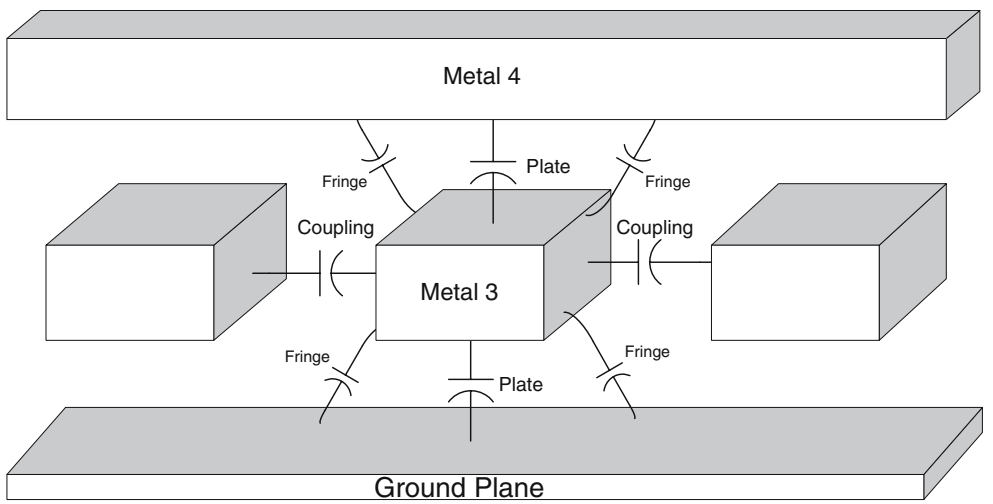


Figure 8. Metal capacitance depends upon physical wire spacing, width and metal layer.

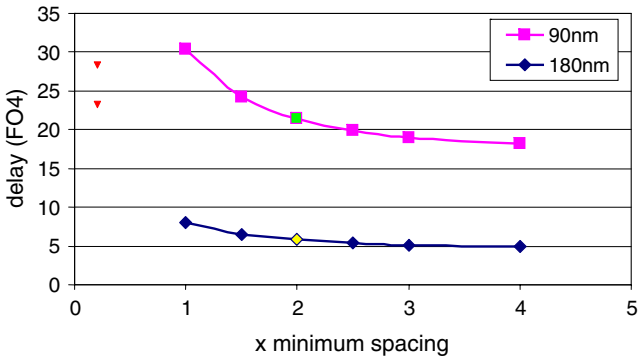


Figure 9. Effect of wire spacing on delay (at minimum width).

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-millimeter. 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. (Area is calculated according to the VPR area model, which computes the area of wide transistors in units of "# of minimum-size transistor areas".)

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 2.5 mm.

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 4 mm should have more than three inverters. Method B is limited to three inverters because it is meant to be relatively quick to compute with only four nested loops. Each additional inverter adds two more nested loops, one for buffer size and

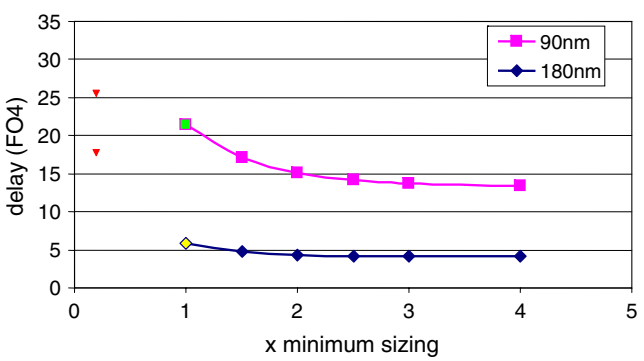


Figure 10. Effect of wire width on delay (at 2× minimum spacing).

Table 3. Solutions found for method A lumped design approach.

Wirelength (mm)	Number of stages (N)	Inverter sizes (×min. size)	HSPICE delay (ps/mm)
Lumped design results (180 nm, 1× spacing, 1× 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 (90 nm, 2× spacing, 2× 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

The reported HSPICE delay includes a 2:1 front-end multiplexer.

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 positions. This was the second factor motivating the development of method C, which is described in the next subsection.

3.4. 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.

Table 4. Method B distributed buffer designs with N=3 inverters using Elmore delay model.

Wirelength (mm)	Best distributed inverter positions (%)	Best distributed inverter sizes	Distributed HSPICE delay (ps/mm)	Lumped HSPICE delay (ps/mm)	Performance difference (%)
Delay-driven results (180 nm, 1× spacing, 1× 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 (180 nm, 1× spacing, 1× 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

Delay of front-end multiplexer is excluded.

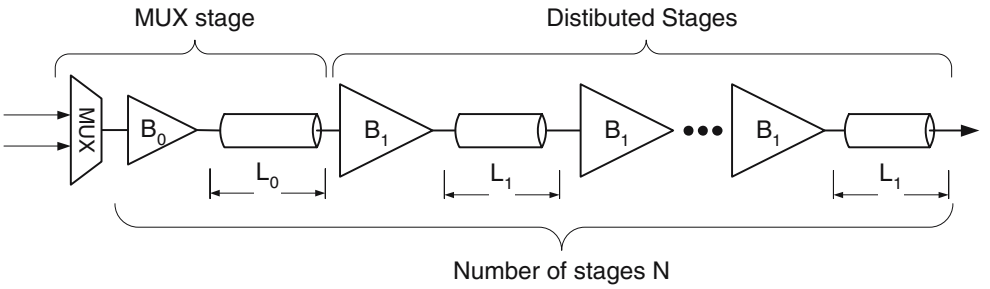


Figure 11. Distributed buffer sizing and spacing problem, simplified.

The method works by first *precharacterizing* two types of circuits using HSPICE, the multiplexer stage and the distributed drive stage shown in Fig. 11 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 buffer size B_1 and the same physical wirelength L_1 to achieve near-minimum delay.

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 Fig. 12. The two adjacent inverters in the multiplexed stage were consistent with method B results. The 2:1 multiplexer is built with minimum size transistors for both PMOS and NMOS.

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

```
for stage_delay = { mux_stage_delay, distrib_stage_delay }
   for W = 0 to 2mm       // the physical wirelength following a circuit
      for B = 1 to 64     // the buffer size of final drive stage
         stage_delay[W,B] = circuit delay using HSPICE
      endfor
   endfor
endfor
```

The results of the precharacterization step can be summarized by two two-dimensional (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 which do *circuit construction*:

```
for N = 1 to 4                  // the number of stages
   for L0 = 0 to 100%           // percent of total wirelength
      L1 = (1.0 - L0) / N       // amount of wire per distrib. stage
      delay = mux_stage_delay[L0] + (N-1)×distrib_stage_delay[L1]
   endfor
endfor
```

An example lookup table is shown in Fig. 13, where the 2D indices are *x*-axis (buffer size) and each curve (wirelength). The result of the lookup table is the *y*-axis (delay). In this figure, the delay is computed for the distributed drive stage with a 180 nm technology using wires of 1× spacing and 1× width.

For fast runtimes, the precharacterization step is limited to two 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], by finding the buffer size that obtains minimum delay and storing both delay and corresponding buffer size at the corresponding wirelength index. These minimum delays and corresponding buffer sizes are shown in Fig. 13 as the vertical (slightly jagged) line near a buffer size of 50×.

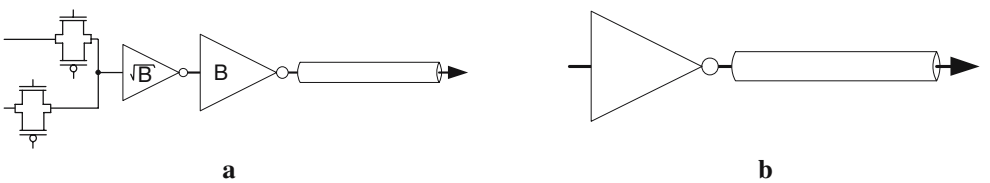


Figure 12. Transistor-level design of mux and distributed drive stages. **a** Multiplexed stage (creates a fast input), **b** distributed drive stage.

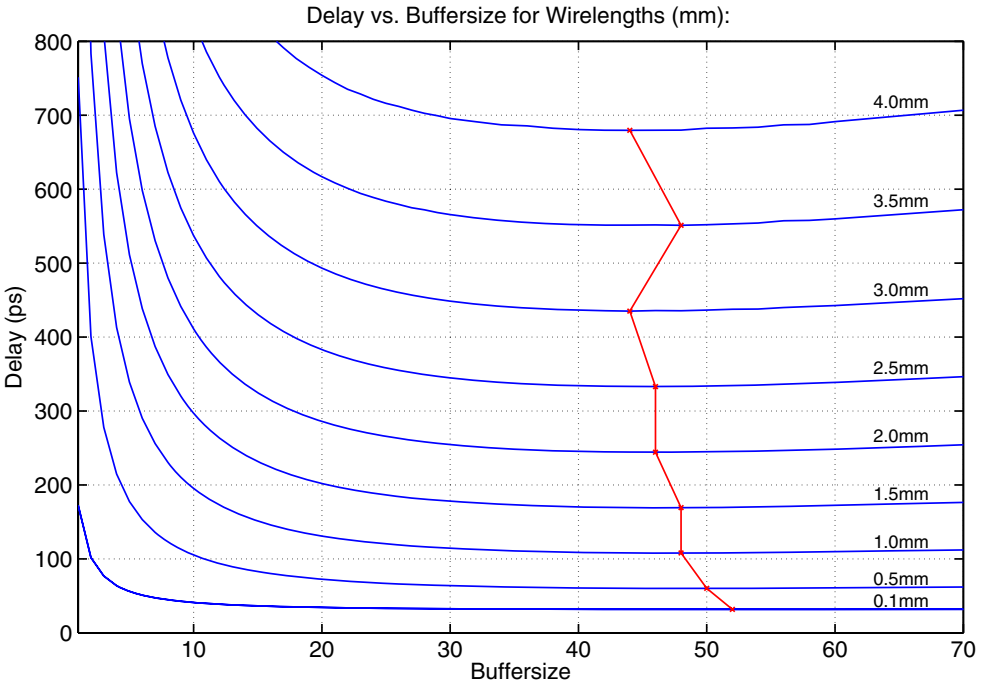


Figure 13. Precharacterization delay lookup table for distributed stage delay in 180 nm using 1× spacing and 1× width.

Since the optimal buffer sizes in the distributed stage are rather large, we relaxed the delay target to 10% above minimum delay and found the corresponding buffer size. This reduces buffer sizes by almost half, from roughly 50× to 30× for 180 nm, but increases delay of the wire itself by less than 10%. The effect of backing off by 10% from minimum delay on buffer size can be seen in Fig. 14. Note especially the much smaller buffer sizes that can be used for very short sections of wire (<1 mm).

Figure 14 also shows the minimum-delay buffer sizes used for the multiplexed stage. Since these sizes are already small enough, and there is only one multiplexer stage per wire, the 10% relaxation technique is not applied there.

Once computed, this lookup-table data can 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 positions for each interconnect wirelength.

The best delay-per-millimeter 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 multiplexed stage data indicates the best FPGA interconnect delay obtained by cascading *only* these mux stages. Table 5 also shows multiplexed signal delays are ∼2× 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.

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 (≤2 mm) 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 >2 mm, method C is superior to method A due to the strong need for distributed buffering.

From Table 6, one might notice that 90 nm has twice the performance of 180 nm. Notice that the 90 nm

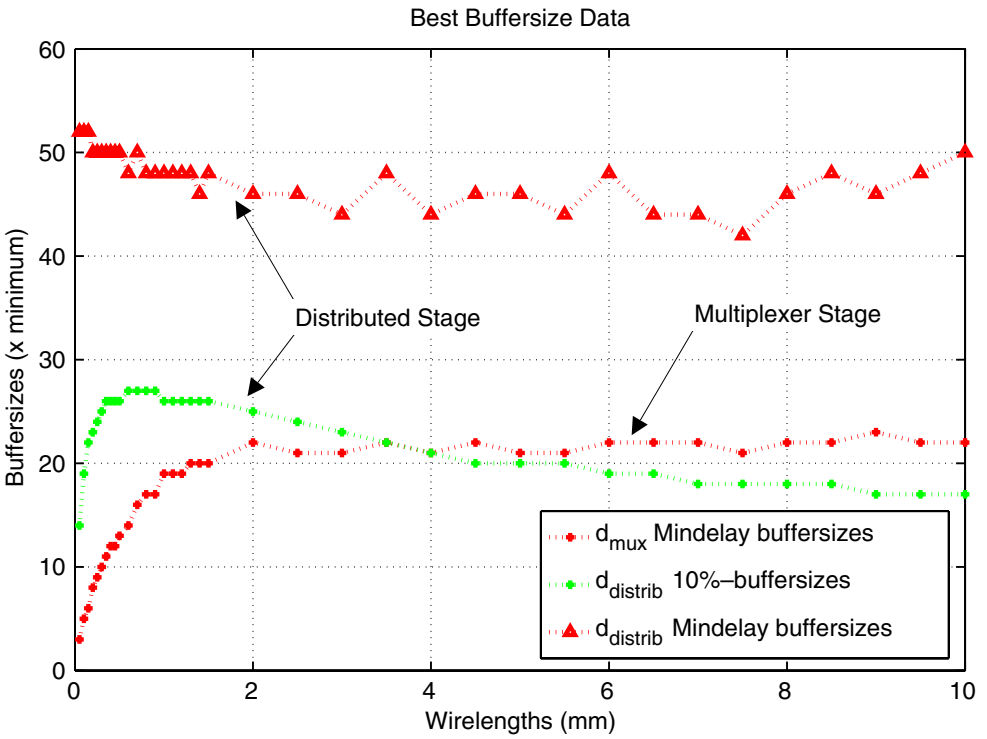


Figure 14. Buffer size lookup table used for different wire lengths.

design is using twice the minimum metal width and spacing but 180 nm is using minimum, i.e., the wires in the two technologies are the same physical width and distance apart, hence they 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 4 mm wire in a 180 nm FPGA must remain 4 mm when the architecture is redesigned for 90 nm. Intuitively, since transistor dimensions are cut in half, a comparable wire only needs to be 2 mm in 90 nm. (This assumes no extra "stuff" is added to a CLB layout tile—no extra logic and no additional routing tracks—which may not be true!) 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.

3.5. *Putting It Together: Multiplexing Interval*

We applied method C to a variety of interconnect lengths from 1 to 10 mm and plotted the delay-per-

Table 5. Best delay-per-millimeter for different stages and wire models extracted from precharacterization data of method C.

Process	FPGA interconnect stage	Delay (ps/mm) 1× spacing, 1× width	Delay (ps/mm) 2× spacing, 2× width
180 nm	Multiplexed	207	138
	Distributed drive[a]	108	69
90nm	Multiplexed	199	131
	Distributed drive[a]	91	58

[a]Also corresponds to best possible ASIC repeater delay (full-swing).

Table 6. Method C distributed buffer designs with one mux stage and $N-1$ identical driver stages using HSPICE precharacterization data.

		1 Multiplexed stage		$N-1$ Distributed stages		
Wire-length (mm)	Number of stages (N)	Driver size B_0 (×min.)	Length L_0 (mm)	Driver size B_1 (×min.)	Length L_1 (mm)	HSPICE delay (ps/mm)
Distributed design results (180 nm, 1× spacing, 1× 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 (90 nm, 2× spacing, 2× 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

The front-end mux is included in the overall delay.

millimeter results in Fig. 15. Since the x-axis represents the planned total physical length of the interconnect wire, we call it 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 1×/1× and 2×/2× wire width/spacing is provided.

Referring to Fig. 15, we see the delay performance of a 4 mm wire in 1×/1×/180 nm is ~155 ps/mm. If the device is rescaled to 90 nm, the wire becomes 2 mm long and delay improves by only 10% to ~140 ps/mm. Increasing width/spacing of the 90 nm wire improves delay to ~105 ps/mm. Alternatively, one might plan to include *longer* architectural wires in the FPGA device at 90 nm. At minimum width/spacing, delay improves to ~115 ps/mm at 4 mm or ~110 ps/mm at 10 mm. 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.

3.6. *Putting It Together: Path Delay Profile*

Method C optimization is driven by HSPICE-calculated delay to the final endpoint. However, in

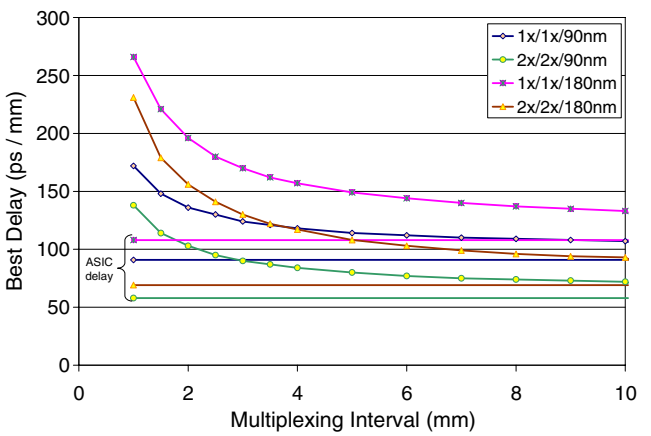


Figure 15. Determining a multiplexing interval for FPGA interconnect.

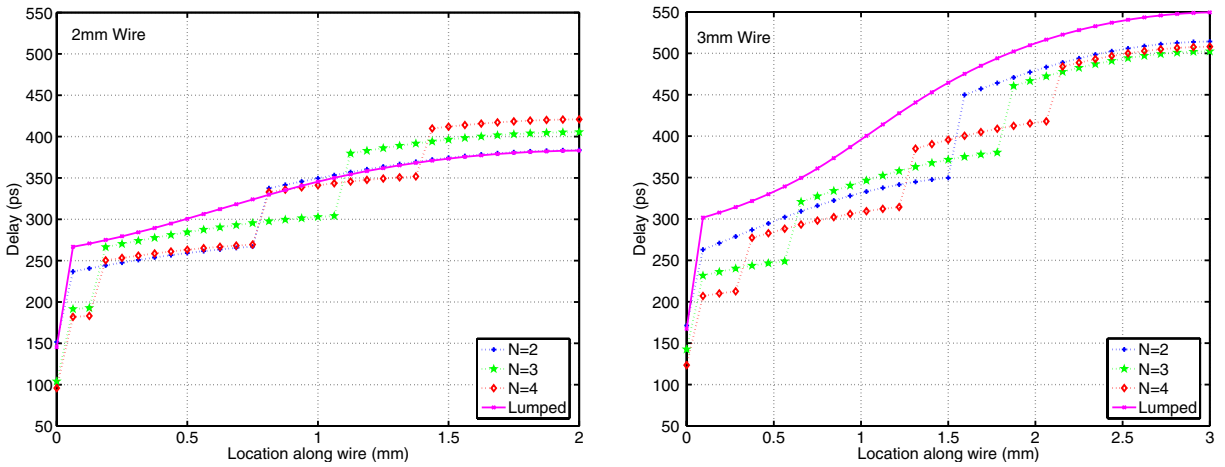


Figure 16. Path delay profiles show advantages of distributed buffering.

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 2 and 3 mm interconnect are plotted in Fig. 16 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 1 mm of a 2 mm interconnect arrive *earlier* than the lumped design, but the points past 1 mm arrive slightly *later*. At a distance of 0.7 mm, distributed is 50 ps (15%) faster and at 2.0 mm it is 40 ps (10%) slower. Overall, it is not clear whether a distributed or lumped buffer design would be faster for 2 mm interconnect after a circuit is placed and routed. This will be investigated in the next section, which shows that distributed buffer design is better.

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

4. 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.

4.1. VPR Changes

Originally, VPR estimates delay to *all points* on a wire as delay to the halfway point using the formula $1/2(R_{\text{wire}}C_{\text{wire}}/2)$. 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.

Table 7. Normalized critical path delay results.

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

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

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 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 positions to the array grid (placing it in either one CLB or the next, not halfway in between). Also, VPR adds a small additional capacitive 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 two-level hybrid multiplexers in HSPICE, extracted delays, and created a parameterized delay model [18]. 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.

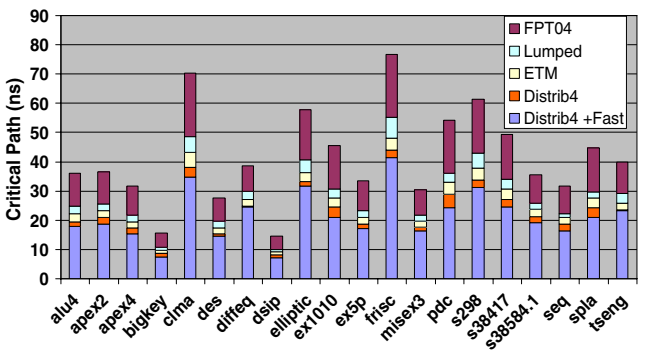


Figure 17. Delay breakdown for a 3.0 mm wire.

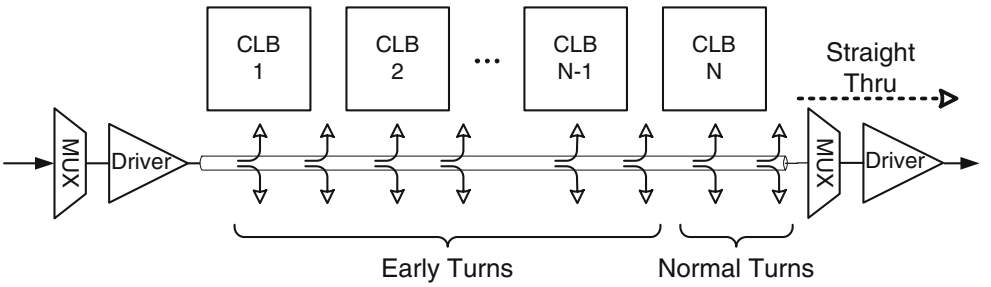


Figure 18. Early turns and normal turns.

4.2. Results

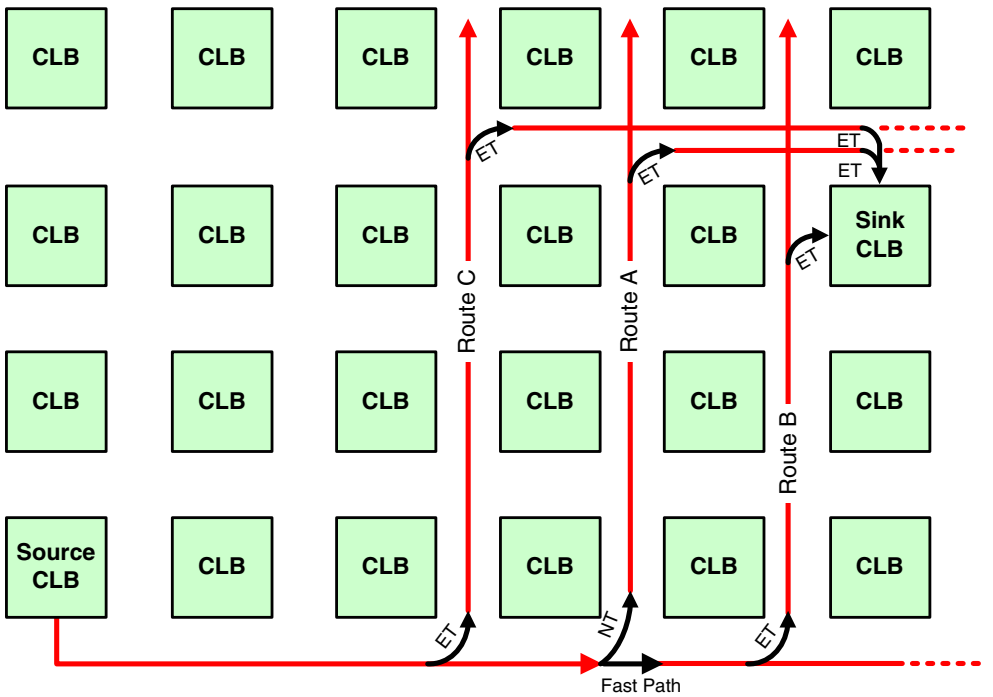
The traditional 20 MCNC benchmark circuits were mapped into four-input LUTs, packed into CLBs containing eight BLEs, and placed once. Then, each placed benchmark was routed several times, each using different interconnect designs. The same channel width is used to route a benchmark across interconnect 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 0.46 mm physical wire lengths, so it is expected to be slower on longer wires.

The performance of several different target interconnect lengths are provided. The 0.5 mm wirelength is assumed to correspond well to a logical wirelength of four CLBs (L4). For the 2 mm and 3 mm wirelengths, a logical wirelength of 16 CLBs (L16) is chosen. Since the MCNC circuits are quite small, they do not fully "exercise" or properly exploit very long wires. To compensate, the benchmark circuits were packed to utilize only one LUT per CLB for L16 logical wirelengths; which spreads 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 long paths), it highlights the benefits of the circuit design which is the focus of this work. Naturally, determining the best mixture of logical

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

Designs	Average total number of turns	Average early turns (%)	Average normal turns (%)	Average straight thrus (%)
Lumped 0.5 mm	7,587	57.3	25.7	16.0
+Fast	7,591	57.3	24.2	17.5
+ETM+Fast	7,698	59.2	20.8	18.9
Lumped 2.0 mm	10,978	87.6	6.9	4.9
+Fast	10,908	88.4	6.4	4.5
+ETM+Fast	11,057	88.4	5.1	5.3
Lumped 3.0 mm	10,983	87.5	6.9	5.0
+Fast	10,913	88.3	6.5	4.5
+ETM+Fast	11,073	88.2	5.2	5.5



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

Figure 19. Early turns alters the ranking of routing preferences.

wire lengths is of great interest, but that effort is best done by FPGA vendors themselves since they have access to several real, large benchmark circuits and complete, detailed modeling information about their own architectures. The focus on this work is to determine the best buffer sizes and positions for wide range of possible scenarios.

From the results in Table 7, the lumped design approach described here improves the FPT04 design of 0.5 mm wires by 10%. Better delay modeling

Table 9. Summary of interconnect computational methods introduced.

Method	Computational complexity	Optimization metric	Limitations
Lumped	$O(N \cdot B)$	HSPICE	Real interconnect is not lumped
Distributed (Elmore)	$O(L^N \cdot B^N)$	Elmore	No front end (multiplexing)
Distributed (HSPICE)	$O(B \cdot L + N \cdot L)$	HSPICE	All distributed stages must be same length

N Number of drive stages, B number of steps in buffer size, L number of steps in wirelength.

(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.

The 2 and 3 mm 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 2 and 3 mm delays. *Even though the raw end-to-end delay performance of 2 mm 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 3 mm design. Overall, the 3 mm wirelength is only 1 ns slower (5%) than the 2 mm wirelength! Also, the 3 mm wirelength achieves an impressive 46% reduction in critical path delay compared to the FPT04 design. A breakdown of delay performance on a circuit-by-circuit basis for the 3 mm wirelength is provided in Fig. 17.

The addition of distributed buffering and improved delay modeling affects decisions made by the router on when/where to turn. Therefore, we decided to investigate the impact of this by analyzing the frequency of turns in the routing solution. Figure 18 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".

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 2 mm 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.

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 Fig. 19. Originally, VPR was unable to distinguish the performance of routes A, B, and C: all paths use three wires and have the same delay, making them equally preferable. 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].

5. Conclusions

This paper has demonstrated three methods for solving the FPGA interconnect buffer spacing and sizing problem: the original lumped method, which uses HSPICE delays and extensively nested loops, a distributed method which also extensively nests loops but uses Elmore delay for faster evaluation at the cost of lower accuracy, and a second distributed method which is both fast and accurate by separating the slow HSPICE-based precharacterization step from the fast computational step used when solving the problem. The three methods are qualitatively compared in Table 9. Method A produces lumped designs with all inverters at one end, while the other two methods provide distributed design solutions. Method C 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 with method C, we demonstrated an average critical path delay reduction of 19% using 0.5 mm wires (logical architectural length 4) compared to previous work. With longer 2 and 3 mm wires, the improvement was even more dramatic at 37 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 is tolerant to small changes in buffer spacing and sizing. This means there is significant room to also address additional optimization goals such as minimum-power switching as well.

Acknowledgments

An earlier version of this work appeared as [20]. Further details can be found in [18]. This research was partially supported by Micronet R&D and NSERC Discovery Grants. Process data and CAD tools were obtained through CMC Microsystems. 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.

References

1. G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and Single-Driver Wiring in FPGA Interconnect," in International 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 International Symposium on FPGAs, Feb. 2006, pp. 113–122.
3. D. Lewis et al, "The Stratix II Logic and Routing Architecture," in International Symposium on FPGAs, Feb. 2005, pp. 14–20.
4. V. Adler and E. G. Friedman, "Repeater Insertion to Reduce Delay and Power in RC Tree Structures," in Conference on Signals, Systems & Computers, Nov. 1997, pp. 749–752.
5. V. Adler and E. G. Friedman, "Uniform Repeater Insertion in RC Trees," IEEE Trans. Circuits Syst. I, vol. 47, no. 10, 2000, pp. 1515–1524.
6. V. Adler and E. G. Friedman, "Repeater Design to Reduce Delay and Power in Resistive Interconnect," IEEE Trans. Circuits Syst. II, vol. 45, no. 5, 1998, pp. 607–616.
7. C. J. Alpert, J. Hu, S. S. Sapatnekar, and C. N. Sze, "Accurate Estimation of Global Buffer Delay Within a Floorplan," IEEE Trans. Comput.-Aided Des., vol. 25, no. 6, 2006, pp. 1140–1146.
8. K. Banerjee and A. Mehrotra, "A Power-Optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs," IEEE Trans. Electron. Devices, vol. 49, no. 11, 2002, pp. 2001–2007.
9. S. Dhar and M. A. Franklin, "Optimum Buffer Circuits for Driving Long Uniform Lines," IEEE J. Solid-State Circuits, vol. 26, no. 1, 1991, pp. 32–41.
10. R. H. J. M. Otten, "Global Wires: Harmful?," in International Symposium on Physical Design, April 1998, pp. 104–109.
11. L. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," IEEE International Symposium on Circuits and Systems, May 1990, pp. 865–868.
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," in Design Automation Conference, 1998, pp. 230–235.
13. V. Betz and J. Rose, "Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect," in IEEE Custom Integrated Circuits Conference, May 1999, pp. 171–174.
14. V. Betz, J. Rose, and A. Marquardt, Architecture and CAD for Deep-Submicron FPGAs: Kluwer, 1999.
15. G. Lemieux and D. Lewis, Design of Interconnection Networks for Programmable Logic, Kluwer, 2004.
16. G. Lemieux and D. Lewis, "Circuit Design of Routing Switches," in International Symposium on FPGAs, Feb. 2002, pp. 19–28.
17. S. Sood, M. Greenstreet, and R. Saleh, "A Novel Distributed and Interleaved FIFO for Source-synchronous Interconnect," VLSI Design and Test Symposium, Goa, 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 International Symposium on FPGAs, February 2005, pp. 21–29.
20. E. Lee, G. Lemieux, S. Mirabbasi, ":Interconnect Driver Design for Long Wires in Field-Programmable Gate Arrays," IEEE International Conference on Field-Programmable Technology, Bangkok, December 2006, pp. 89–96.

Edmund Lee received the B.A.Sc. degree in computer engineering from University of Toronto, in 2003 and the M.A.Sc degree in computer engineering from University of British Columbia in 2006. He worked as a co-op intern at Actel (2001–2002) for 16 months, and a research intern at Intel (2004) for 6 months. Mr. Lee co-authored a paper that won the Best Paper Award at the *IEEE International Conference on Field-Programmable Technology* in 2004. He is presently working at the Altera Toronto Technology Centre.

Guy Lemieux received the B.A.Sc degree from the division of engineering science at the University of Toronto, and the M.A.Sc. and Ph.D. degrees in electrical and computer engineering at the University of Toronto. Dr. Lemieux's research interests include computer-aided design algorithms, VLSI and SoC circuit design, FPGA architectures, and parallel computing. His specialization is in interconnection network design and routing algorithms. In 2003, he joined the Department of Electrical and Computer Engineering at The University of British Columbia, Canada, where he is an Assistant Professor. Dr. Lemieux. He is co-author of the book *Design of Interconnection Networks for Programmable Logic* and he co-authored a paper that won the Best Paper Award at the *IEEE International Conference on Field-Programmable Technology* in 2004.

Shahriar Mirabbasi received the B.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran in 1990 and the M.A.Sc and Ph.D. degrees in electrical and computer engineering from University of Toronto, Canada in 1997 and 2002, respectively. From May to August 1997, he was with Gennum Corporation, Burlington, ON, Canada working on the system design of cable equalizers for serial digital video and HDTV applications. During January 2001 to June 2002, he worked at Snowbush Microelectronics, Toronto, Canada, on the design and test of high-speed mixed-signal CMOS integrated circuits including ADC and serializer/deserializer blocks. In August 2002, he joined the Department of Electrical and Computer Engineering of the University of British Columbia, Vancouver, BC, Canada, where he is currently an Associate Professor. His current research interests include analog, mixed-signal, and radio-frequency integrated circuit and system design for high-speed wireless and wireline data communication applications, wireless sensor networks, and biomedical implants.