The Speed of Diversity: Exploring Complex FPGA Routing Topologies for the Global Metal Layer

Oleg Petelin and Vaughn Betz
Department of Electrical and Computer Engineering
University of Toronto, Toronto, Canada
{opetelin, vaughn}@eecg.toronto.edu

***Abstract*—The rapid growth of wire RC delay with technology scaling has put increasing pressure on FPGA architects to make more efficient use of the different layers available in the metal stack. While commercial FPGA architectures have implemented the majority of inter-logic-block wiring on the lower metal layers and a small fraction of wires on the least-resistive upper metal layers, published explorations have largely ignored the question of how to exploit the different layers of the metal stack, focusing instead on very simple interconnect topologies and physical models. We generate VPR architectures and detailed area and delay models at the 22nm node and present enhancements to VPR that enable us to describe and evaluate complex interconnect topologies. We use our new architectures and tool enhancements to explore complex interconnect patterns suitable for modern unidirectional architectures and suggest topologies to connect wires on the semi-global and global metal layers. The proposed topologies improve the critical path routing delay by 17% compared to architectures with no global layer wires, and by 5-13% compared to architectures with global layer wires using the default VPR switch pattern.**

I. Introduction

The routing architectures of Field-Programmable Gate Arrays (FPGAs) consist of wires and programmable switches that, with suitable programming, allow the logic resources to be connected to meet the needs of any application circuit. For a typical FPGA design, most of the delay and 50% or more of the area [1] [2] is due to this programmable interconnect, so its optimization is a priority for FPGA architects. Moreover, interconnect RC delay has increased rapidly with process scaling, particularly for the thin wires at the bottom of the metal stack [3] [4], and the challenges of poor interconnect resistance scaling require FPGA architects to use the metal layers with greater efficiency. Interconnect architecture is a complex topic, and includes many interacting architectural decisions such as wire electrical characteristics, wire segment length and the switch pattern between wires. While each of these topics has been studied before on its own, e.g. in [5] [6] [7], the interaction between switch patterns and wire types has not been explored in published literature. In this paper we develop new tools and models that allow us to investigate complex interconnect topologies and switch patterns across different wire types. We then explore the impact of unidirectional (direct-drive) wiring on switch pattern, as well as suitable interconnect hierarchies to take advantage of scarce, fast wiring on the upper metal layers.

A cross section of the metal stack used in Intel's 14nm process is shown in Figure 1 [8]. The metal layer stack consists of a large number of wires with a small cross-section close to the silicon and a smaller number of wider and taller wires on the upper metal layers. The thin wires close to the silicon are naturally useful for the multitude of short-distance connections, while the larger less-resistive wires further up lend themselves to long-distance connections, clocking, and power distribution.

While commercial architectures have implemented a small fraction of their inter-logic-block wires on the highest metal layers, published explorations of interconnect topologies have largely ignored the question of how to exploit the electrical characteristics of the different layers available. In addition, most published explorations of switch patterns have targeted only very simple routing architectures with all length 1 routing wires and bidirectional switches. In contrast, modern commercial devices use a mix of wire lengths and have moved to unidirectional (mux-based) switches which place significant restrictions on the switch pattern. In this paper we seek to fill a gap in published explorations of FPGA interconnect:

- We use the FPGA modelling tool COFFE [9] along with HSPICE to extract accurate delay and area parameters at the 22nm process node for use in VPR architecture files. Challenges such as increasing wire RC delay are most pronounced in the latest process nodes, so we believe modelling such an advanced node is crucial for relevant results.
- We make multiple enhancements to the CAD tool VPR in order to support the complex routing architectures we wish to explore. These enhancements include a new switch block format that allows us to succinctly specify virtually any interconnect topology in the VPR architecture file, as well as an improved router lookahead that can take advantage of different wire

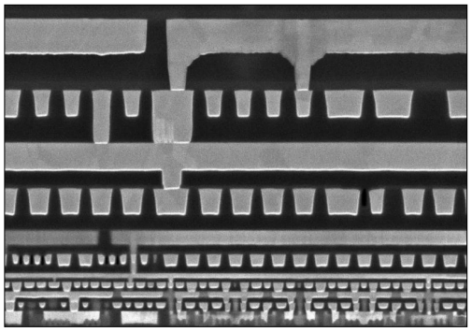

Fig. 1: Intel's 14 nm process metal stack [8].

types connected in complex ways.

- We investigate the impact of unidirectional architectures on switch block topology and show that the restrictions imposed by unidirectional wires actually help to permute the switch pattern, increasing the number of FPGA tracks that signals using restrictive switch blocks can reach.
- We use our 22 nm VPR architecture files and VPR enhancements to explore complex interconnect patterns and suggest switch topologies to exploit wires on the top-most metal layers. Specifically, we explore hierarchies to connect fast wires on the upper metal layers to the rest of the routing fabric, and show an improvement of 5-13% over the default VPR switch pattern using the same wire type mix.

The rest of this paper is organized as follows. Section II discusses background and prior works; Section III outlines our 22nm architectures and Section IV discusses our enhancements to VPR. Section V explores complex routing topologies and Section VI concludes.

II. Background

A. Routing Elements of Island-Style FPGAs

Figure 2 illustrates the basic routing elements of an island-style architecture. The figure shows a single FPGA tile, many of which are stamped across the chip. Parameters of the tile routing elements are defined similarly to [10].

- W is the channel width and refers to the number of wire segments in a channel. The length of a wire segment is defined as the number of tiles that the wire spans. Figure 2 shows wire segments of length 1.
- Connection blocks (CBs) interface wire segments with logic blocks (LBs) through programmable switches. Input and output CB flexibilities, $F_{c,in}$ & $F_{c,out}$, represent the fraction of wires that a pin of an LB connects to in the routing channel.
- Switch blocks (SBs) provide programmable connections between different routing wire segments through buffered, non-tristatable muxes. The wire segments are thus unidirectional and can be driven at the start of the segment only [5]. The switch block flexibility F_s is the number of other wires to which a wire segment connects inside the switch block.

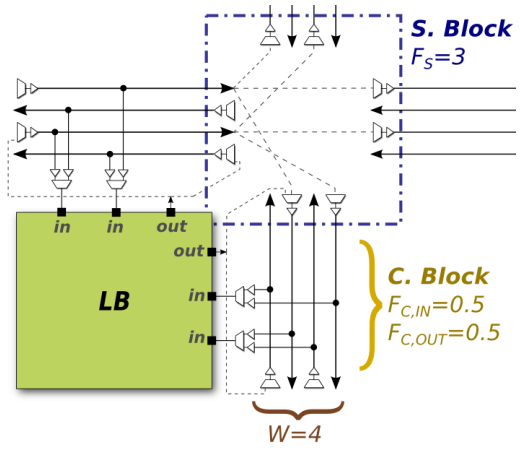

Fig. 2: Key components of an island-style routing architecture using unidirectional wires.

B. Interconnect and Scaling

The International Technology Roadmap for Semiconductors (ITRS) classifies the types of wires in the metal stack into the M1, intermediate, semi-global and global layers [11]. Semi-global and global layers have wires with dimensions that are 2x and 4x+ of the intermediate layer respectively. M1 and intermediate-layer wires have the same dimensions, but differ somewhat in electrical characteristics.

While interconnect scaling has traditionally maintained an approximately-constant delay for wires on the lowest metal layers[1], manufacturing challenges and nano-scale effects have increased the resistance of copper interconnect by a significant amount. The impact of poor wire resistance scaling has already been felt in commercial devices [4], and ITRS projects that at the current rate, interconnect will become a major delay bottleneck in the near future [11]. With interconnect scaling posing an increasingly difficult challenge, it is important that FPGA architects make effective use of the relative advantages offered by the different metal layers.

C. Complex Routing Architectures

Popular commercial architectures from Xilinx and Altera [12] [13] use multiple types of wires to route signals. While to our knowledge Xilinx has not published full details of their recent routing architectures, Altera's Stratix series takes advantage of the relative strengths of different metal layers by implementing the majority of inter-LB routing on a slower, more accessible metal layer, while a low-resistance metal layer towards the top of the metal stack is used to implement a small number of long (length 16+) wires [13]. On the uppermost metal layers, layout restrictions of deep via stacks prevent wire connections to logic blocks and other wires at every tile, restricting vias to every four LBs [14]. Recently the Stratix V architecture [1] has moved a portion of short y-directed wires higher up in the metal stack, though the uppermost metal layers are still used only for long wires.

Published explorations for designing heterogeneous interconnect topologies have been limited. Switch block patterns like the subset [10], universal [15] and Wilton [7] were developed for length-1 bidirectional architectures and had a significant impact on routability. These patterns were then adopted for longer wires and even a mix of wire lengths. The Imran switch block [16] has been the only published topology that specifically exploits long wires, but it still assumes bidirectional wiring. Bidirectional architectures have largely been supplanted by unidirectional architectures that reduce routing area and critical path delay [5]. Since a unidirectional wire can be driven at only one point, such architectures place major restrictions on the switch pattern, which have not been evaluated in published studies.

Studies of wirelength in [2] used the Wilton switch block with no adjustments to optimize the switch pattern for the variety of wire lengths. It also used a bidirectional routing architecture and a much older (.35 um) process technology with lower resistance wires, making its conclusions questionable for the latest FPGAs.

Overall, combined studies of wire segment length, wire type and switch pattern are needed for unidirectional topologies, and in Section V we explore the impact of unidirectional

[1]Decreasing wire cross section increases resistance, but decreasing wire length maintains an approximately-constant delay.

wires on switch pattern, as well as interconnect hierarchies to take advantage of fast wires on the global metal layer.

D. CAD Tools and Algorithms

CAD systems such as VTR (which includes the tool VPR) [17] are able to explore FPGA design trade-offs by running a full elaborate/synthesize/pack/place/route flow over a set of benchmark circuits. This flow outputs metrics such as critical path delay and the area dedicated to routing resources, which help to evaluate design decisions. While VPR can model any set of wire lengths in its architecture files, it includes only 3 predefined switch blocks (subset, Universal and Wilton). In Section IV we describe modifications to VPR that allow the specification of arbitrary switch blocks from which VPR can then construct a routing graph describing the entire chip.

The routing algorithm of VPR operates on a graph of nodes representing FPGA resources by routing net connections from source nodes to targets [2]. A router lookahead is an important component of the timing-driven routing algorithm because it provides estimates of the remaining delay (and possibly congestion) to reach targets from intermediate nodes. The lookahead is vital to keep the router runtime low since it enables the router to explore the graph in a targeted manner as opposed to exploring the graph with a breadth first search which can be slower by more than an order of magnitude [18].

The router lookahead requires some form of knowledge about the interconnect topology in order to estimate the remaining delay and congestion to reach the sink since an inaccurate estimate can degrade runtime, performance or both. The VPR router takes the following approach [2]: having found itself on a given kind of node (for example, a slow length-4 wire) as the router looks for a routing path, the VPR lookahead estimates the delay by assuming that the smallest possible number of wires of the same type (slow length-4 wires) will be used to reach the sink. For the complex interconnect topologies that we wish to explore this strategy is not sufficient: a slow length-4 wire may connect to a fast length-16 wire on the global metal layer downstream, and correctly estimating the remaining delay can improve the quality of results.

The router lookahead proposed in Independence [19] is able to target complex interconnect topologies as it makes no assumptions about the FPGA routing architecture. The architecture-adaptive lookahead of Independence uses K-means clustering to group routing nodes which have a similar delay to reach a small number sinks in the routing graph. Sample routing from the clustered "supernodes" to a subset of sinks in the graph is then used to build lookup tables that are accessed during routing to estimate the remaining delay (all nodes in a supernode share a table entry). While this lookahead algorithm is very general, the lookup tables constructed through this method can have a prohibitively large memory footprint, especially as the number of supernodes increases for large FPGAs.

In Section IV we propose a new lookahead algorithm for the VPR timing-driven router that is able to take advantage of complicated wiring patterns while maintaining a low compute and memory requirement.

III. Architectures in 22nm

The 22nm FPGA architecture on which we build our interconnect explorations is based on the flagship VPR 7.0 40nm architecture and uses a logic block similar to that of popular commercial devices [1] [20]. This architecture is summarized in Table I and we use it for most of our results. However, in Section V-C we also check the accuracy of our main results using a logic block architecture similar to [21] which has 4-input LUTs and no internal crossbars; this architecture is summarized in Table II.

TABLE I: Logic architecture for most of our interconnect explorations.

LB Size	Ten 6-input Fracturable LUTs
LB Input Crossbar	Full Crossbar
LB Output Crossbar	None
LB Internal Feedback	Through Input Crossbar
DSP Elements	36x36 Fracturable Multipliers
Memories	32Kb Block RAMs
$F_{c,in}$	0.1
$F_{c,out}$	0.1

TABLE II: Architecture used to check sensitivity of main interconnect results.

LB Size	Eight 4-input LUTs
LB Input Crossbar	None
LB Output Crossbar	None
LB Internal Feedback	None
DSP Elements	36x36 Fracturable Multipliers
Memories	16Kb Block RAMs
$F_{c,in}$	0.2
$F_{c,out}$	0.2

TABLE III: Metal stack data from the 2014 entry of the ITRS 2011 interconnect report.

Metal Layer	Half-Pitch (nm)	Aspect Ratio	R (ohm/um)	C (fF/um)
Intermediate	24	1.9	54.825	0.175
Semi-Global	48	2.12	7.862	0.215
Global	96	2.34	1.131	0.250

To have timing and area parameters representative of the 22nm node, the transistor-sizing tool COFFE [9] was used to size a number of architectures over different wire segment lengths, metal layers and routing flexibilities. The metal stack data used is shown in Table III and was extracted from the 2011 ITRS interconnect report [3] from which the 2014 entry was chosen to represent the 22nm node.

By default COFFE uses the intermediate metal layer to implement the wiring inside the logic block and the semi-global layer to implement the general routing outside of it. We wish to explore wire length and topology choices for general interconnect on the global metal layer, so to extract delay and area parameters from COFFE we did the following. For each combination of {*semi-global, global*} metal layer and length-{*1, 2, 4, 8, 16*} general routing wires we used COFFE to perform transistor sizing on an architecture with a channel width of 300 containing only one of the aforementioned wire type combinations. For each combination we also varied the connection block and switch block flexibilities in order to capture the delay impact of different routing mux sizes. Area and delay parameters extracted from COFFE were then used for VPR architecture files. Representative parameters for the different wire types are shown in Table IV. Longer wires and wires on the global metal layer present a larger capacitive load

TABLE IV: Extracted 22nm delay and area data for different wire types. Each entry shows data for the (*semi-global / global*) metal layers. VPR and COFFE measure area in Minimum-Width Transistor Areas (MWTAs).

	L1	L2	L4	L8	L16
Routing Switch Output Resistance (ohm)	1110 / 840	740 / 500	520 / 300	520 / 230	350 / 200
Routing Switch Intrinsic Delay (ps)	57 / 50	64 / 52	80 / 60	132 / 83	235 / 107
Routing Mux Pass Gate Area (MWTAs)	1.5 / 1.5	1.5 / 1.7	1.7 / 1.7	1.5 / 1.7	1.5 / 2.0
Routing Buffer Area (MWTAs)	11 / 13.6	15 / 19	23 / 27	25 / 39	38 / 50
Wire Resistance Per Tile (ohm)	232 / 34				

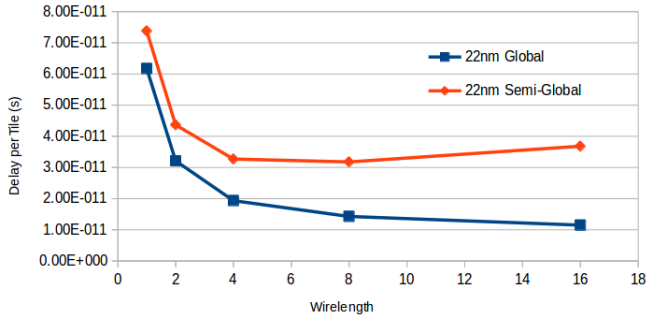

Fig. 3: Delay per tile versus wire length on the semi-global and global layers. Delay per tile is the total delay through a loaded wire segment (including the driver delay) divided by the segment length.

and so require bigger, less resistive routing switches[2]. The last row of Table IV shows that wires on the global metal layer have significantly lower resistance per tile, where the dimensions of a tile were reported by COFFE to be $30\mu m$ by $30\mu m$.

Figure 3 contrasts the delay difference between the semi-global and global metal layers for multiple wire lengths. The increasing RC delay of advancing technology nodes clearly shows its impact – in contrast with some of the earliest routing architecture studies [2], semi-global length-8 and length-16 wires are now too resistive to provide a delay benefit over length-4's without removing connections to SBs or CBs.

IV. Tool Enhancements

Our exploration of complex interconnect patterns required multiple enhancements to the VPR CAD tool which we describe here. These enhancements will be contributed back into VPR and will be documented in the VPR user manual.

A. Enhanced Switch Block Descriptions

VPR 7.0 can specify three kinds of switch blocks in the architecture file – subset, universal and Wilton. While these switch patterns have a significant impact on routability for bidirectional topologies, they do not have a clear analogue for unidirectional wires. VPR adapts these switch patterns to unidirectional topologies by snapping switch block connections to the nearest wire start point, which results in a somewhat ambiguous pattern. More importantly, the switch blocks used by VPR do not allow us to specify some of the complex interconnect topologies that we wish to explore. For example, later in this paper we explore a routing architecture with "regular" and "fast" wires.. The regular wires are driven by both regular and fast wires, but a signal can find its way onto a fast wire only through the output connection block or another fast wire. While our new switch block description format is able to describe architectures like the one above, the switch block patterns in VPR 7.0 would not be able to specify this kind of interconnect hierarchy and would connect the wire types in a semi-random manner.

Our switch block description format, which can be specified in the VPR architecture file, is a generalized version of the permutation function tables described in [22]. We can specify a mathematical permutation function that determines how two different wire types defined in the architecture file can connect at a switch block. For example, one can specify how a signal taking an east-north turn at a switch block will connect from one of the midpoints of a length-4 wire to the start point of a length-16 wire. This description format allows us to succinctly specify any switch block we wish to explore in a well-defined manner, including switch blocks where different wire types interconnect with varying flexibilities.

B. Enhanced Connection Block Descriptions

We have enhanced VPR's connection block format to allow varying connection block flexibility based on the type of wire to which a pin connects. For example, this enhancement allows us to describe architectures where fast wires on the global metal layer can be driven from a connection block with a higher flexibility than the rest of the routing but have no direct connection to input pins, forcing routed paths to reach inputs through regular wires on the semi-global layer instead.

C. Varying Delays Based on Mux Fan-in

The size and topology of the buffered muxes used to connect routing wires is influenced by the mux fan-in. Both COFFE and VPR use a 1-level topology to implement smaller muxes and a 2-level topology to implement muxes with larger fan-ins (VPR 7 uses mux fan-in and topology to account for area differences between muxes). The signal delay through a mux grows with fan-in, and it is important to capture this effect in our interconnect explorations. We did so by enhancing the VPR architecture files to accept a list of *(fanin, delay)* pairs for each routing switch in the architecture file. During the generation of the VPR routing resource graph, interpolation in this delay table is used to choose the delay of each multiplexer, as only at that point is the precise fan-in of each mux known. As mentioned in Section III, COFFE was used to extract timing parameters for switches driving each kind of wire type we explored over a range of fan-in values.

D. Enhanced Router Lookahead

We propose a new lookahead that is well-suited to the regularity of island-style FPGA architectures. Our lookahead is more general than the lookahead used in VPR and has a very low memory requirement compared to that of Independence.

Figure 4 summarizes the lookahead generation algorithm. This algorithm computes look-up tables by running Dijkstra's algorithm sorted on delay from a small subset of wires belonging to each *wire type* that has been defined in the VPR

[2]The routing switch intrinsic delay was modelled to include the wire resistance and capacitance (but not connection block & switch block loading). This still allows us to accurately model the effects of capacitive loading from additional switches, but simplifies parameter extraction from COFFE HSPICE files.

```
1: function GENERATE_LOOKAHEAD(routing resource graph)
2:     REF_X=3, REF_Y=3, M = 10
3:     lookahead_map = NULL
4:
5:     For each wire_type in the VPR architecture file{
6:       For chan_type in {x-chan, y-chan}{
7:         For M wires at REF_X, REF_Y{
8:           n = get_routing_resource_graph_node( wire )
9:           priority_queue = NULL
10:          priority_queue.push(n)
11:
12:          //Run Dijkstra's algorithm sorted on path delay
13:          While priority_queue not empty{
14:            n = get_lowest_delay_node(priority_queue)
15:            if (n is input pin &&
16:            n.x >= REF_X && n.y >= REF_Y){
17:              dX = n.x - REF_X
18:              dY = n.y - REF_Y
19:              old_T = get_path_delay(
20:                lookahead_map[wire_type][chan_type][dX][dY])
21:
22:              // lookahead_map to contain smallest-delay entry
23:              if (old_T == UNDEFINED ||
24:              old_T > n.path_delay){
25:                lookahead_map[wire_type][chan_type][dX][dY] =
26:                  add_entry(n.path_delay, n.base_path_cost)
27:              }
28:            }
29:            add_neighbors_by_path_delay(n, priority_queue)
30:          } //While
31:        } //For M wires
32:      } //For chan_type
33:    } //For each wire_type
34:
35:    return lookahead_map
36: end function
```

Fig. 4: Algorithm to generate look-up tables to be used by the router lookahead.

architecture file (i.e. the algorithm is run from a few slow length-4's, a few fast length-16's, or any other wire types that were defined). The search is performed at one reference coordinate from both x-directed and y-directed channels and notes two pieces of information each time an input pin is encountered above and to the right of the starting tile: the *path delay* and the *base path cost* to arrive at that pin – these are the two costs (timing and resource) that VPR needs to estimate during routing. Note that the search is performed away from the edge of the chip to avoid fringe effects, and we assume symmetry by only searching up and to the right of the starting coordinate. This search records the minimum *path delay* and associated *base path cost* required to reach each coordinate relative to the starting tile[3].

Our lookahead takes advantage of the regularity of the island-style FPGA architecture to build small look-up tables that can be rapidly indexed by the router. For example, to estimate the delay to travel 3 tiles in the x direction and 5 tiles in the y direction starting from an x-directed wire of type *len4*, the router would lookup the *base_path_cost* and *path_delay* information under index *[len4][x-chan][3][5]*.

[3]We also tested approaches that record multiple *path delay* and *base path cost* entries at each dX and dY and then boil them down to a single representative entry by taking the average, median or geometric mean. However, we found that such methods, which potentially overestimate path cost, can be detrimental to VPR's routing time.

TABLE V: Memory, compute requirements and generality of different router lookaheads for a 200x200 island-style FPGA.

Router Lookahead Algorithm	Memory Footprint	Lookup Table Compute Time	Generality
VPR	0	0	✓
Independence	> 6GB	Not Listed	✓✓✓
Proposed	< 10MB	30s	✓✓

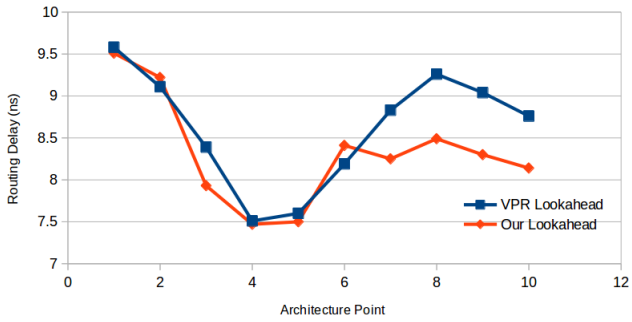

Fig. 5: Average critical path routing delay over 9 largest VTR benchmarks, for two lookaheads, over 8 distinct architectures.

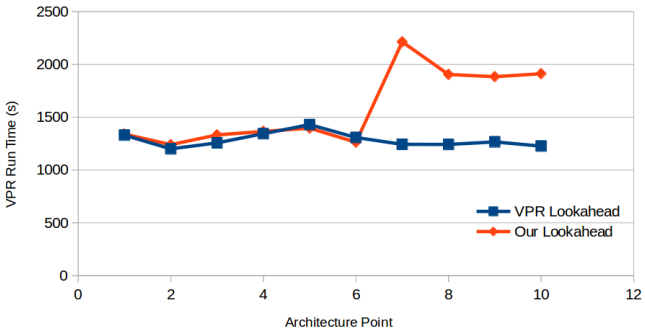

Fig. 6: Average VPR runtime over 3 largest VTR benchmarks, for two lookaheads, over 8 distinct architectures.

Table V compares the compute and memory requirements of our lookahead to that of VPR and Independence. While not as general as Independence, our lookahead achieves a good trade-off point due to its much more manageable memory footprint. With a computational complexity of $Nlog(N)$ (N is the number of nodes in the routing graph), our lookahead scales well to very large chips and adapts to arbitrary wire type mixes and switch patterns, assuming only that the chip is translationally invariant (which island-style FPGAs are).

Figures 5 and 6 compare our lookahead with VPR's over two simple, four moderately complex and four very complex architectures. The first two points represent simple architectures with only one type of wire available. The next four architectures (points 3-6) have 85% of wires on the semi-global layer and 15% on the global layer; the global wires can only be driven from block output pins and can only drive semi-global wires. The last four points (points 7-10) represent architectures where the global wires (which comprise 15% of the channel width) can be driven, and can only drive, a small fraction of the regular wires on the semi-global layer. Our lookahead has quality and runtime equivalent to the VPR lookahead for simple and moderately complex architectures (points 1-6), and has significantly better result quality for complex architectures at the expense of route time (points 7-10). Despite the longer

TABLE VI: *Benchmarks for evaluating routing delay. †Benchmarks for evaluating routability. Data from [17].

Circuit	#6-LUTs	Min W (len-4 wires)
mcml*	99700	144
LU32PEEng*	75530	204
bgm*†	30089	168
stereovision2*	29849	172
LU8PEEng*†	21954	136
stereovision0*†	11462	78
stereovision1*†	10366	120
blob_merge*†	6016	100
mkDelayWorker32B*†	5580	110
or1200†	2963	90
boundtop†	2921	72
sha†	2212	64
raygentop†	2134	74
mkSMAdapter4B†	1977	80

runtime for the last four architecture points, our lookahead adapts well to their complex and irregular structure, and we use it for our interconnect explorations.

V. Architecture Explorations

A. Benchmarks and Methodology

We use VPR along with the VTR benchmark set [17] to evaluate routability and delay; the benchmarks used are summarized in Table VI. In Section V-B we evaluate routability by finding W_{min}, the minimum routable channel width of each circuit. In Sections V-B and V-C, critical path routing delay (the portion of the critical path through the inter-LB routing) is evaluated at a constant channel width of W=300, which provides some extra flexibilty over what is required by our most complex benchmark circuit. We consider timing results in a fixed channel width the most representative as all commercial FPGA families ultimately choose a single channel width, and a channel width of 300 aligns well with popular commercial FPGA devices [23] with a logic block of ten fracturable 6-LUTs similar to the one we use (Table I).

In our architecture explorations we have treated the fastest metal layers as a scarce resource – in addition to being used for fast general routing wires, these layers are utilized for power, ground and clocking. Furthermore, the via stack required to connect from the highest metal layers to silicon presents significant layout challenges since vias must traverse through all the intermediate metal layers. Commercial architectures have used 10-20% of the available channel width for fast long wires, and have restricted wires on upper metal layers to have vias approximately every four logic blocks [14] [23].

For our delay experiments we use architectures with a channel consisting of 300 unidirectional wires where 15% of the channel width can be used for wires on the global metal layer. To reflect the layout difficulties of deep via stacks, wires on the global metal layer are allowed to have connection block and/or switch block connections only once every four tiles.

B. Effect of Unidirectional Wires on Switch Pattern

In this section we look at the best single wirelength and switch block topology to be used for the semi-global metal layer in our subsequent explorations. Commercial architectures have typically based their routing around a shorter (approximately length-4) wirelength and a switch block topology

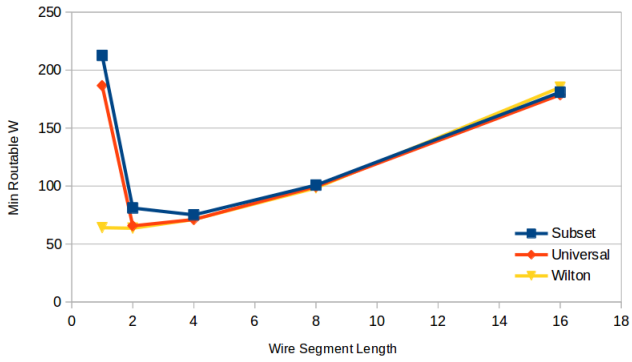

Fig. 7: Routability of switch block patterns converges for longer unidirectional wires.

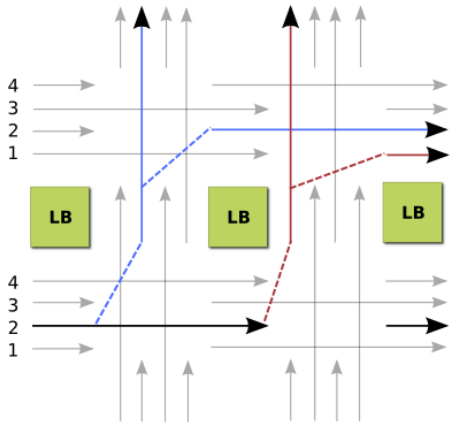

Fig. 8: Restrictions of unidirectional wiring make it impossible to create traditional low-routability patterns like the subset at wire lengths greater than 1.

that allows signals to have access to a diverse set of tracks during the course of a route; we validate this choice of semi-global wirelength at the 22nm node and make some interesting observations about switch block topology in unidirectional architectures.

In Figure 7 we evaluate the routability of the subset, universal [15] and Wilton [7] switch blocks implemented with unidirectional wires of varying length on the semi-global layer. As expected, longer wire lengths are generally less routable because they decrease the granularity with which signals can utilize the routing resources.

With regard to switch blocks, at length-1 the Wilton topology is much more routable. While the universal and subset switch blocks restrict a signal to a small subset of possible wires, the Wilton switch block permutes the switch pattern so that signals are able to find their way onto virtually any track in the FPGA[4]. However, at longer wire lengths the routability

[4]The low connection block flexibility in our architectures (see Table I) contributes to the significantly lower routability of the subset and Universal switch blocks at the length-1 point. Signals from an output pin are confined to a small fraction of available wires in the FPGA due to the restrictive nature of these switch blocks. The wires that an output pin connects to would hence not be guaranteed to line up with the switches connected to the input pins of I/O, RAM or DSP blocks (which do not have an input crossbar in our architecture) causing some nets to be unroutable even without congestion. Increased F_c would likely improve the overall area usage of these patterns, but they would still remain inferior choices to the Wilton switch block at length 1.

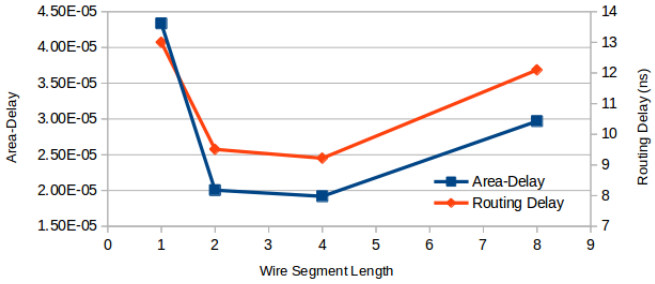

Fig. 9: Area-delay (blue) and critical path routing delay (red) of architectures with a single type of semi-global wire using the Wilton switch block.

of the different switch block patterns begins to converge.

In unidirectional architectures wires can only be driven at their start point which eliminates some of the area required for bidirectional routing [5]. We observe that the restriction of connecting only to the start points of unidirectional wires permutes the switch block connections beyond what the switch block pattern specifies for wire lengths greater than 1. In Figure 8 we illustrate an architecture with length-2 wire segments where a signal on track 1 can make its way onto tracks 1 and 2 by taking turns through the routing (regardless of switch block), doubling the number of tracks a subset switch block would be able to reach compared to a length-1 architecture. With a subset switch block, length-L wires increase the number of tracks that a signal can reach by a factor of L, and the restriction of connecting to wire start points can further help stagger the output connection block pattern. The routability of switch block topologies therefore converges at longer wirelengths – an interesting observation about unidirectional routing that to our knowledge has not been previously published. In our multi-wirelength explorations we choose the Wilton switch block but focus on how different routing resources connect instead of the specific switch pattern that implements the connection.

Figure 9 verifies that length-4 wires still achieve the best area-delay tradeoff at 22nm, and length-4's form the basis of our complex architecture explorations.

C. Complex Routing Topologies

As shown in the previous section, unidirectional wires act to permute the switch block pattern beyond what is specified and, at moderate and long wire lengths, analogues of low-routability patterns like the subset are difficult to create. In this section we explore high-level hierarchies to connect fast wires on the global metal layer with the rest of the routing. Figure 10 illustrates the topologies; the thick arrows represent wire types and the thin arrows represent how each wire type connects to other wires and FPGA blocks. Each connection between wire types is implemented with a Wilton switch pattern adapted to that length of unidirectional wire but, as shown in Section V-B, other switch permutations can likely be used without significantly affecting the result. The topologies explored represent three kinds of wire type hierarchies:

- *No distinction between wire types.* The *default VPR 7* switch pattern connects wires using the Wilton switch block, snapping connections to the start points of unidirectional wires. With this pattern, some wires connect to other wire types, and some wires connect to the same wire type, producing a somewhat irregular interconnect where connections between wire types are not guaranteed.
- *Isolated wire types.* The (a) *On-CB/Off-CB* topology of Figure 10 does not implement connections between wire types, and wires connect only within their own type and to FPGA blocks. This topology was also used in [24] to evaluate a mix of regular wires and fast (widely-spaced) wires.
- *Connected wire types.* Topologies (b), (c), (d) and (e) in Figure 10 have guaranteed connections between wire types, and also vary the available connections between wire types and FPGA blocks.

As mentioned in Section V-A, each topology allocates 15% of the available channel width to wires on the global metal layer, and connections to/from global-layer wires are restricted to one in every four tiles to reflect the layout difficulties of deep via stacks. Each topology is evaluated over a number of global-layer wire lengths and is compared to an architecture using the default VPR 7 switch pattern with the same wire mix, as well as to an architecture using only length-4 semi-global wires. The delay and per-tile routing area results are shown in Figure 11.

With reference to Figures 10 and 11, the topology results are discussed below:

(a) *On CB, Off CB*: In this topology global wires are only driven by output pins and can only drive input pins and other global wires; essentially the global wires form a completely distinct routing network from the regular (semi-global) wires. Figure 11 shows that length-4 global wires achieve a slight critical path routing delay reduction over architectures with only semi-global wires, while longer wire lengths don't provide much benefit. While this topology worked well in [24] (13% speedup), it does not work well with global-layer wire segments, which can not be used efficiently due to the layout difficulties of deep via stacks.

(b) *On CB, Off SB*: Global wires are only driven by output pins and can only drive other global/semi-global wires. As in (a), each output pin still has a dedicated connection to a fast wire on the global layer, but the ability of global wires to "jump down" to semi-global wires adds an extra level of flexibility to the routing and allows signals to use the global wires with greater efficiency. For the same wire type mix, this topology reduces critical-path routing delay by 4-12% compared to the default VPR switch pattern.

(c) *On CB, Off CB/SB*: Global wires are driven only by output pins and drive global/semi-global wires and LB input pins. Compared to (b), the ability of global wires to drive some LB input pins allows some nets to be routed using the fast global layer wires exclusively which further reduces the routing delay (5-13% compared to the default VPR switch pattern) at the slight expense of extra per-tile routing area.

(d) *On SB, Off SB*: With this topology global wires can drive, and can be driven from a fraction of the semi-global wires. Compared to (c), length-16 global wires provide a greater delay benefit since their connection with semi-global wires provides an extra degree of flexibility. On the other hand, length-4 and length-8 wires have a higher routing delay compared to (c) because signals seeking to use the global layer must first traverse semi-global wires.

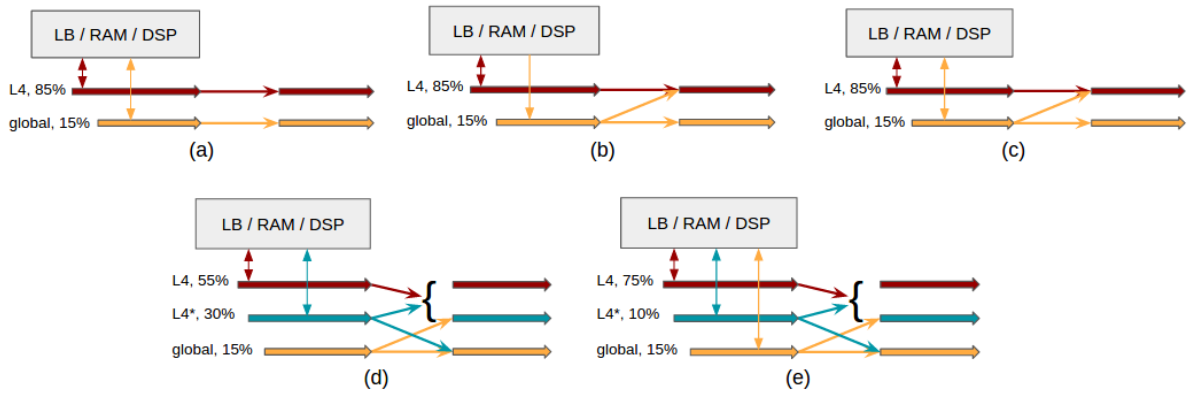

Fig. 10: Different routing topologies with regular and fast wires. Topology names refer to the connectivity of global wires. (a) On CB, Off CB topology (b) On CB, Off SB topology. (c) On CB, Off CB/SB topology. (d) On SB, Off SB topology; only a fraction of regular L4 wires can drive global wires. (e) On CB/SB, Off CB/SB topology.

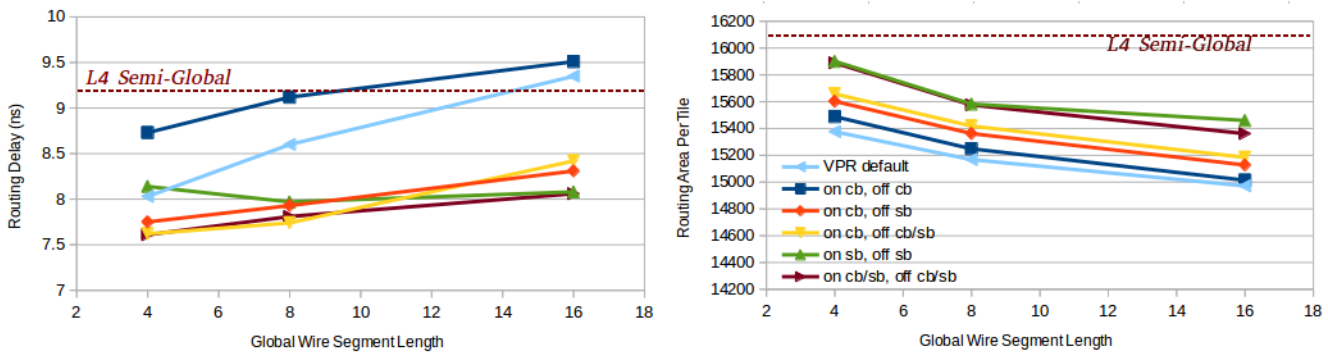

Fig. 11: (a) Critical path routing delay for different interconnect topologies and global metal layer wire lengths. (b) Per-tile routing area (in MWTAs) for different interconnect topologies and global metal layer wire lengths.

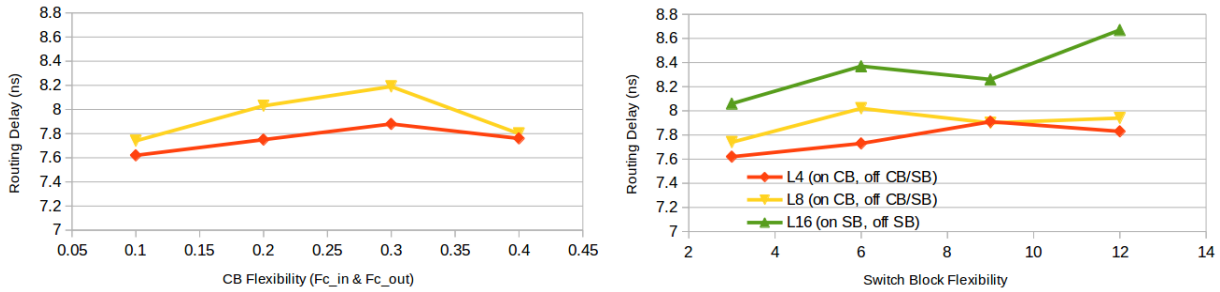

Fig. 12: (a) Sweeping input/output connection block flexibility for best interconnect topologies at each global metal wire length (best architecture with global length 16 wires does not have global CB connections and is not included here). (b) Sweeping switch block flexiblity for best interconnect topologies at each global metal wire length.

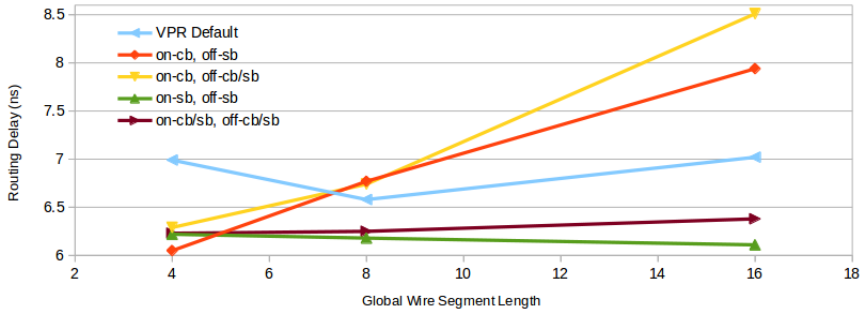

Fig. 13: Critical path routing delay results for a 4-input LUT logic block architecture without internal crossbars.

(e) *On CB/SB, Off CB/SB*: Global wires can be driven and can drive both LB pins and a fraction of the available semi-global wires. This topology provides no delay benefits over (c) for shorter wire lengths and no benefits over (d) for length-16 wires.

The results show the importance of connections between different wire types, and all topologies with guaranteed connections between wire types performed relatively well. However, the exact choice of topology depends on global-layer wire length. Two important observations are:

- Shorter global-layer wires are best driven directly from the output pins of FPGA blocks to give signals immediate access to fast routing resources. On the other hand, longer unidirectional wires have fewer start points in each channel segment, and should be driven from wires on the semi-global layer to provide a greater degree of flexibility. Lastly, global-layer wires of all lengths should drive regular wires on the semi-global layer; the restriction of having via connections every four tiles makes it impractical for global wires to drive pins directly through the input connection block without traversing the semi-global layer wires first
- While popular commercial architectures use long wire lengths (length 16+) on the least-resistive metal layers, our explorations show that with the appropriate topology, shorter wire lengths can be made surprisingly fast. Just as short wires are more routable, shorter wires on the global metal layer allow a greater number of signals to take advantage of the fast routing resources, potentially speeding-up a larger number of timing-critical connections.

Figure 12 shows CB and SB flexibility sweeps for the best topology at each global wirelength – increased F_c and F_s appear to add capacitive loading without improving critical path delay through extra routing flexibility.

The sensitivity of architecture explorations to experimental setup is well known [25], and in addition to averaging our results over multiple benchmark circuits, we have also repeated our topology explorations using a logic block architecture similar to [21], which has eight 4-input LUTs and no internal crossbars. Figure 13 shows the critical path routing delays for this architecture over a subset of medium-size VTR benchmarks. While our main observations remain the same, one important difference is present: the lack of internal crossbars places an increased emphasis on routing flexibility, and both length-8 and length-16 global-layer wires must now be driven by regular wires on the semi-global layer; only length-4 wires see a delay benefit using *On-CB* topologies.

Lastly, we suspected that length-16 global-layer wires may further reduce critical path delay when larger benchmarks are used. We checked our results using only the three largest VTR benchmarks but did not observe improvement in the relative delay performance of longer wire lengths.

VI. Conclusion

FPGA interconnect architecture has always been a complex topic and the rapidly increasing RC delay of scaling interconnect motivates FPGA architects to use the available metal layers more efficiently. In this paper we have sought to fill a gap in published explorations of interconnect topology by enhancing architecture models and tools, and exploring interconnect hierarchies to take advantage of fast global-layer routing across different wire lengths.

While traditional switch block patterns like the subset, universal and Wilton had significant routability differences when bidirectional architectures were used, we have shown in Section V-B that unidirectional wires act to permute switch patterns beyond what a traditional switch block specifies. With the exception of short wire lengths, the requirement of connecting to wire start points places a limit on how restrictive a switch pattern can be made, and analogues of low-routability patterns like the subset switch block are difficult to create. Therefore, rather than focusing on the detailed switch pattern, in Section V-C we explored high-level hierarchies to connect fast global-layer wires to the rest of the routing, and suggest two observations for exploiting this fast but scarce metal layer:

- For greatest delay benefits, global wires of all lengths should drive regular semi-global layer wires. As well, shorter wires are best driven directly from the connection block while long wires are best driven from regular semi-global layer wires.
- With the right interconnect topology, shorter wires on the global metal layer can be surprisingly fast. Compared to long wires, the higher routing flexibility of shorter wires allows more signals to use the fast routing resources, which can further decrease critical path routing delay.

Our best topologies follow the above rules and improve the critical path routing delay by 17% compared to architectures with no global layer wires, and by 5-13% compared to architectures with global layer wires using the default VPR switch pattern.

In our explorations we have allocated 15% of the available channel width to global-layer wires and have restricted global-layer wire connections to one in four logic blocks. While these choices are reasonable [14] [23] [24], it would be interesting to investigate the impact of via depopulation and percentage of global-layer wires on the choice of interconnect topology, and future work will explore this fully.

Acknowledgement

This work is funded by Lattice Semiconductor and the NSERC/Altera Industrial Research Chair in Programmable Silicon. We would like to extend our thanks to Jun Zhao, Duan-Ping Chen, Brad Sharpe-Geisler and David Rutledge for their helpful discussions and suggestions. Computations were performed on the gpc supercomputer at the SciNet HPC Consortium. SciNet is funded by: the Canada Foundation for Innovation under the auspices of Compute Canada; the Government of Ontario; Ontario Research Fund - Research Excellence; and the University of Toronto.

References

[1] D. Lewis *et al.*, "Architectural Enhancements in Stratix V," in *FPGA*, 2013, pp. 147–156.

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

[3] International Technology Roadmap for Semiconductors, "2011 Report, Interconnect Chapter."

[4] G. Yeap, "Smart Mobile SoCs Driving the Semiconductor Industry: Technology Trend, Challenges and Opportunities," in *IEDM*, 2013.

[5] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and Single-Driver Wires in FPGA Interconnect," in *FPT*, 2004, pp. 41–48.

[6] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density," in *FPGA*, 1999, pp. 59–68.

[7] S. J. E. Wilton, "Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memory," Ph.D. dissertation, University of Toronto, 1997.

[8] Intel Corporation, "Advancing Moore's Law on 2014," 2014.

[9] C. Chiasson and V. Betz, "COFFE: Fully-Automated Transistor Sizing for FPGAs," in *FPT*, 2013, pp. 34–41.

[10] J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," *JSSC*, pp. 277–282, 1991.

[11] International Technology Roadmap for Semiconductors, "2013 Report, Interconnect Chapter."

[12] Xilinx, "UltraScale Architecture and Product Overview."

[13] D. Lewis *et al.*, "The Stratix II Logic and Routing Architecture," in *FPGA*, 2005, pp. 14–20.

[14] D. Lewis et al., "The Stratix Routing and Logic Architecture," in *FPGA*, 2003, pp. 12–20.

[15] Y.-W. Chang and D. F. Wong, "Universal Switch Modules for FPGA Design," *ACM TODAES*, vol. 1, no. 1, pp. 80–101, 1996.

[16] M. Imran Masud, "FPGA Routing Structures: A Novel Switch Block and Depopulated Interconnect Matrix Architectures," Master's thesis, University of British Columbia, 1999.

[17] J. Luu *et al.*, "VTR 7.0 : Next Generation Architecture and CAD System for FPGAs," *ACM TRETS*, vol. 7, no. 2, 2014.

[18] J. Swartz, V. Betz, and J. Rose, "A Fast Routability-Driven Router for FPGAs," in *FPGA*, 1998, pp. 140–149.

[19] A. Sharma and S. Hauck, "Accelerating FPGA Routing Using Architecture-Adaptive A* Techniques," in *FPT*, 2005, pp. 225–232.

[20] S. Chandrakar, D. Gaitonde, and T. Bauer, "Enhancements in UltraScale CLB Architecture," in *FPGA*, 2015, pp. 108–116.

[21] Lattice Semiconductor, "ECP5 and ECP5-5G Family Data Sheet," 2016.

[22] G. Lemieux and D. Lewis, *Design of Interconnection Networks for Programmable Logic*. Springer New York, 2004.

[23] K. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz, "Titan: Enabling Large and Complex Benchmarks in Academic CAD," in *FPL*, 2013.

[24] V. Betz and J. Rose, "Circuit Design, Transistor Sizing and Wire Layout of FPGA Interconnect," in *CICC*, 1999, pp. 171–174.

[25] A. Yan, R. Cheng, and S. Wilton, "On the Sensitivity of FPGA Architectural Conclusions to Experimental Assumptions, Tools, and Techniques," in *FPGA*, 2002, pp. 147–156.